TUGAS AKHIR – EE 184801

PENERAPAN BLOCKCHAIN DAN KRIPTOGRAFI UNTUK KEAMANAN DATA PADA JARINGAN SMART GRID

Hafizh Fianto Putra
NRP 07111440000145

Dosen Pembimbing
Dr. Ir. Wirawan, DEA.
Prof. Ir. H. Ontoseno Penangsang, M.Sc., Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

FINAL PROJECT – EE 184801

THE IMPLEMENTATION OF BLOCKCHAIN AND CRYPTOGRAPHY FOR DATA SECURITY IN SMART GRID NETWORK

Hafizh Fianto Putra
NRP 07111440000145

Supervisors
Dr. Ir. Wirawan, DEA.
Prof. Ir. H. Ontoseno Penangsang, M.Sc., Ph.D.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

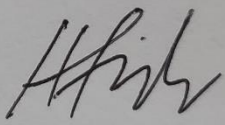
PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Penerapan Blockchain dan Kriptografi untuk Keamanan Data pada Jaringan Smart Grid**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Januari 2019



Hafizh Fianto Putra
NRP. 07111440000145

PENERAPAN BLOCKCHAIN DAN KRIPTOGRAFI UNTUK KEAMANAN DATA PADA JARINGAN SMART GRID

TUGAS AKHIR

Diajukan Untuk Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Telekomunikasi Multimedia
Departemen Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Ir. Wirawan, DEA.	Prof. Ir. Ontoseno P., M.Sc., Ph.D.
NIP. 196311091989031011	NIP. 194907151974121001

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI ELEKTRO
DEPARTEMEN TEKNIK ELEKTRO

SURABAYA
JANUARI 2019

PENERAPAN BLOCKCHAIN DAN KRIPTOGRAFI UNTUK KEAMANAN DATA PADA JARINGAN SMART GRID

Hafizh Fianto Putra
07111440000145

Dosen Pembimbing I : Dr. Ir. Wirawan, DEA.
Dosen Pembimbing II : Prof. Ir. H. Ontoseno Penangsang, M.Sc., Ph.D.

ABSTRAK

Smart city adalah sebuah konsep kota yang mengandalkan teknologi *Internet of Things* (IoT) untuk membantu meningkatkan kualitas kota tersebut. *Smart grid* adalah salah satu komponen dari *smart city* yang memperkenalkan komunikasi dua arah antara pelanggan dengan perusahaan penyedia listrik. Salah satu masalah yang dapat terjadi pada jaringan *smart grid* adalah data pelanggan yang jatuh ke pihak yang tidak bertanggungjawab karena saluran transmisi yang tidak aman. Oleh karena itu, penggunaan *blockchain* dan kriptografi dapat diterapkan pada jaringan *smart grid* untuk menyelesaikan masalah tersebut. *Blockchain* memberikan keamanan tambahan untuk penyimpanan data pada pusat pengatur jaringan dan kriptografi memberikan kerahasiaan dan autentikasi pada pertukaran data dalam jaringan. Sistem yang baru diciptakan ini harus bisa menyesuaikan frekuensi pengambilan data pada jaringan *smart grid*. Penelitian ini bertujuan untuk mengetahui waktu komputasi yang dibutuhkan dari kedua proses tersebut dengan menyusun suatu simulasi jaringan *smart grid*. Simulasi dalam penelitian ini dilakukan dengan menggunakan aplikasi MATLAB dan tersusun dari dua program utama, yaitu program kriptografi menggunakan algoritma RSA dan program pembuatan *blockchain* sederhana. Hasil yang didapatkan menunjukkan bahwa rentang waktu komputasi RSA pada suatu jaringan *smart grid* lebih cepat dari frekuensi pengambilan data yang ditentukan, yaitu sebesar satu menit. Sementara itu, rentang waktu penyusunan *blockchain* dengan ketentuan tertentu tidak dapat memenuhi persyaratan yang sama.

Kata Kunci : *Blockchain*, Kriptografi, MATLAB, RSA, *smart grid*

Halaman ini sengaja dikosongkan

THE IMPLEMENTATION OF BLOCKCHAIN AND CRYPTOGRAPHY FOR DATA SECURITY IN SMART GRID NETWORK

Hafizh Fianto Putra
07111440000145

Supervisor I : Dr. Ir. Wirawan, DEA.
Supervisor II : Prof. Ir. H. Ontoseno Penangsang, M.Sc., Ph.D.

ABSTRACT

A smart city is a city concept that relies on the Internet of Things (IoT) technology to help improve the quality of the city. A smart grid is a component of the smart city that introduces two-way communication between customers and electricity companies. One problem that can occur in smart grid networks is customer data that falls to irresponsible parties due to unsafe transmission lines. Therefore, the use of blockchain and cryptography can be applied to smart grid networks to resolve these problems. Blockchain provides additional security for storing data in the central network regulator and cryptography provides confidentiality and authentication for data exchange on the network. This newly created system must be able to adjust the frequency of data retrieval on the smart grid network. This study aims to determine the computational time needed from the two processes by compiling a simulation of smart grid networks. The simulation in this study is done by using the MATLAB application and is composed of two main programs, namely cryptographic programs using the RSA algorithm and a simple blockchain creation program. The results obtained show that the RSA computing time span in a smart grid network is faster than the specified data retrieval frequency, which is equal to one minute. Meanwhile, the time frame for the preparation of the blockchain with certain conditions cannot meet the same requirements.

Keywords : Blockchain, cryptography, MATLAB, RSA, smart grid

This page is intentionally left blank

KATA PENGANTAR

Dengan mengucap puji syukur kepada Allah SWT atas limpahan rahmat dan berkat-Nya, saya dapat menyelesaikan buku tugas akhir ini dengan judul:

"PENERAPAN BLOCKCHAIN DAN KRIPTOGRAFI UNTUK KEAMANAN DATA PADA JARINGAN SMART GRID"

Tugas akhir ini disusun sebagai salah satu persyaratan dalam menyelesaikan studi pada bidang studi Telekomunikasi Multimedia di departemen Teknik Elektro, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam kesempatan ini, saya ingin menyampaikan rasa terima kasih kepada semua pihak yang telah mendukung saya selama proses menyelesaikan tugas akhir ini, khususnya kepada:

1. Ibu Fitri dan Bapak Sunu, kedua orang tua saya yang telah memberikan banyak dukungan, mendoakan, dan merestui langkah saya.
2. Bapak Wirawan dan Bapak Ontoseno, kedua dosen pembimbing tugas akhir saya yang telah memberikan banyak ilmu, pengarahan, dan bimbingan selama penyelesaian tugas akhir ini.
3. Teman-teman angkatan 2014 yang telah memberikan semua kenangan dan bantuan selama masa perkuliahan saya di ITS.
4. Negara Jepang yang telah memberikan kenangan dan pengalaman berharga bagi saya selama pertukaran pelajar, juga sebagai pemicu semangat saya untuk bisa meraih kesuksesan.

Saya menyadari adanya kekurangan dan keterbatasan selama penyusunan laporan tugas akhir ini. Oleh karena itu, saya selalu terbuka terhadap kritik dan saran untuk membantu meningkatkan kualitas penelitian saya.

Semoga buku ini dapat memberikan informasi dan manfaat bagi semua kalangan, khususnya mahasiswa Teknik Elektro dan bidang studi yang sejenisnya. Saya juga berharap buku ini mampu memberikan kontribusi terhadap perkembangan keilmuan, khususnya di bidang telekomunikasi.

Surabaya, Januari 2019
Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Listrik merupakan salah satu komoditas yang paling sering digunakan di dunia saat ini. Penggunaan listrik telah menghasilkan berbagai jenis teknologi baru selama beberapa tahun terakhir. Untuk memenuhi kebutuhan masyarakat dunia yang semakin tinggi, banyak inovasi yang juga terlahir di bidang kelistrikan. Salah satu gagasan yang ditemukan adalah sistem *smart grid*.

Smart grid adalah sebuah sistem komunikasi dua arah antara pelanggan dengan perusahaan penyedia listrik. *Smart grid* sejatinya adalah salah satu komponen dari *smart city*, kota yang menerapkan *Internet of Things* (IoT) untuk meningkatkan kualitas kota dan rakyat yang tinggal di kota tersebut. *Smart grid* dapat memiliki fleksibilitas dalam integrasi beberapa komponen dalam jaringan distribusi listrik.

Masalah yang dapat terjadi pada *smart grid* adalah data pelanggan yang jatuh ke pihak yang tidak bertanggung jawab karena saluran transmisi yang tidak aman. Data yang didapatkan tersebut dapat disalahgunakan sehingga merugikan pelanggan maupun perusahaan penyedia listrik. Pelanggan bisa saja membayar biaya lebih dari yang seharusnya atau perusahaan penyedia listrik yang merugi karena pemasukan dari pelanggan kurang dari pengeluaran yang dilakukan [1].

Penggunaan *blockchain* dan kriptografi dapat diterapkan pada jaringan *smart grid* untuk menyelesaikan masalah tersebut. *Blockchain* merupakan sebuah rekaman autentik dari seluruh aktivitas yang terjadi dalam suatu jaringan, tersebar di seluruh komponen jaringan, dan saling terkait dalam suatu rangkaian blok data. *Blockchain* memberikan keamanan tambahan untuk penyimpanan data pada pusat pengatur jaringan. Di sisi lain, sistem kriptografi juga diterapkan untuk keamanan dan autentikasi data yang dikirimkan sehingga data tersebut hanya bisa diakses oleh pengirim dan penerima, serta keasliannya juga terjamin.

Sistem yang baru diciptakan tersebut harus bisa menyesuaikan frekuensi pengambilan data pada jaringan *smart grid*. Penelitian ini bertujuan untuk mengetahui waktu komputasi yang dibutuhkan dari kedua proses tersebut dengan menyusun suatu simulasi jaringan *smart grid*. Simulasi dilakukan dengan menggunakan aplikasi MATLAB dan tersusun dari dua program utama, yaitu program kriptografi menggunakan algoritma RSA dan program pembuatan *blockchain* sederhana.

1.2 Rumusan Masalah

Dari latar belakang yang telah disebutkan, masalah yang dirumuskan untuk dapat diselesaikan dalam penelitian ini adalah:

a. Bagaimanakah model yang terbaik untuk jaringan *smart grid* yang dapat bebas dari segala tindak kecurangan?
b. Apakah *blockchain* dan kriptografi dapat menjadi solusi untuk menyelesaikan masalah keamanan pada jaringan *smart grid*?
c. Apakah variabel yang terdapat pada *blockchain* dan kriptografi dapat memengaruhi sistem jaringan *smart grid*?
d. Apakah simulasi jaringan *smart grid* yang dihasilkan dapat langsung diterapkan secara nyata tanpa menimbulkan permasalahan lainnya?

1.3 Batasan Masalah

Penelitian ini memiliki beberapa batasan atau asumsi untuk mempermudah penyelesaian masalah yang telah disebutkan. Batasan-batasan tersebut adalah:

a. Saluran transmisi dianggap bersifat sempurna atau tidak menimbulkan galat pada data yang ditransmisikan
b. Kriptografi yang digunakan adalah RSA orisinal
c. Jaringan bersifat statis atau jumlah klien telah ditetapkan sebelumnya
d. Simulasi tidak dilakukan menggunakan perangkat *smart meter* yang sebenarnya, tetapi hanya menggunakan laptop dan komputer
e. Kode program yang dibuat bersifat *single-threading* atau hanya menjalankan satu baris program dalam satu waktu

1.4 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

a. Menemukan model yang terbaik untuk jaringan *smart grid* yang dapat bebas dari segala tindak kecurangan
b. Membuktikan bahwa *blockchain* dan kriptografi dapat menjadi solusi untuk menyelesaikan masalah keamanan pada jaringan *smart grid*
c. Mengetahui pengaruh yang dapat ditimbulkan oleh perubahan variabel yang terdapat pada *blockchain* dan kriptografi

d. Menghasilkan simulasi jaringan *smart grid* yang dapat langsung diterapkan secara nyata tanpa adanya masalah

1.5 Metodologi

Dalam penelitian ini, pekerjaan yang dilakukan dapat dipecah menjadi empat bagian utama, yaitu: Studi Literatur, Pembuatan Simulasi, Pengujian Program, dan Analisis Data. Penjelasan dari tiap kegiatan tersebut adalah sebagai berikut:

a. Studi Literatur

Studi literatur diperlukan untuk membahas beberapa topik khusus seperti RSA dan *blockchain*. Selain itu, studi yang dilakukan juga mencakup tentang pencarian kondisi terkini melalui *paper* yang telah dipublikasikan.

b. Pembuatan Simulasi

Simulasi dari penelitian ini menggunakan program MATLAB dan potongan kode program disusun untuk bisa menyelesaikan permasalahan yang telah ditentukan.

c. Pengujian Program

Setelah program dapat berjalan dengan baik, program akan diuji dengan beberapa variabel yang diubah-ubah. Hal tersebut dimaksudkan untuk mencari hasil yang terbaik. Tahap berikutnya, potongan program tersebut akan disatukan dan dijalankan sesuai desain jaringan yang telah ditentukan.

d. Analisis Data

Setelah program dijalankan pada perangkat, hasil yang diperoleh akan dianalisis. Analisis data dilakukan untuk mempermudah penarikan kesimpulan.

1.6 Sistematika Penulisan

Pembahasan Tugas Akhir ini dibagi menjadi lima bab dengan sistematika sebagai berikut :

Bab 1 Pendahuluan

Bab ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika pembahasan, dan relevansi Tugas Akhir ini.

Bab 2 Tinjauan Pustaka

Bab ini berisi penjelasan mengenai teori penunjang yang mendukung dalam pembahasan Tugas Akhir ini, seperti *smart grid* dan RSA.

Bab 3 Perancangan Sistem

Bab ini berisi pembahasan potongan kode program yang digunakan untuk simulasi dan rencana implementasi program pada suatu jaringan *smart grid*.

Bab 4 Analisis dan Pembahasan Data

Bab ini berisi data hasil simulasi program pada perangkat. Analisis dan pembahasan dari data yang didapatkan juga terdapat pada bagian ini.

Bab 5 Penutup

Bab ini berisi kesimpulan dan saran dari hasil pengujian program yang telah dilakukan.

1.7 Relevansi / Manfaat

Hasil yang diperoleh dari penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

a. Pemahaman terkait topik kriptografi dan aplikasinya pada dunia nyata.
b. Penyelesaian masalah yang memerlukan kemampuan berpikir logika dan menerapkannya pada sebuah program.
c. Lebih banyak penelitian lintas bidang studi seperti *smart grid* yang dilakukan untuk masa yang akan datang.
d. *Blockchain* dapat menjadi topik baru di dalam lingkungan sarjana ITS dan menjadi pembuktian bahwa *blockchain* bukan merupakan topik yang terlalu susah atau kompleks untuk dipelajari.

BAB 2
TINJAUAN PUSTAKA

2.1 *Smart Grid*

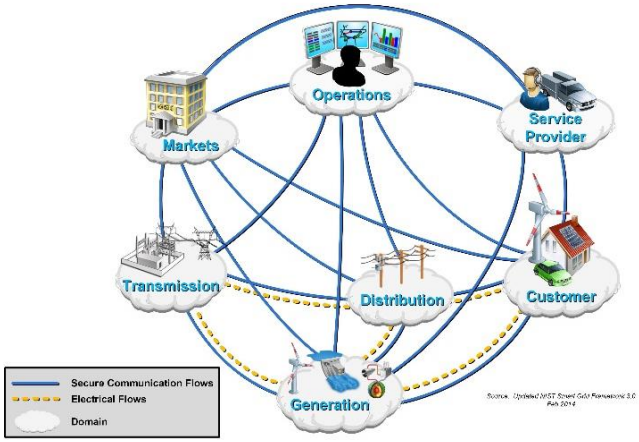
Smart grid adalah sebuah jaringan yang menggunakan teknologi informasi pada kelistrikan agar efisien, dapat diandalkan, dan aman [2]. Pada jaringan *smart grid*, akan ada pertukaran data secara dua arah antara pengguna listrik dengan perusahaan penyedia listrik. Contoh data yang dapat diambil dari meteran listrik pengguna adalah nilai tegangan, arus, dan *power factor* [3].

Tabel 2.1 menjelaskan mengenai informasi dasar dari beberapa data yang pernah diambil dan dipublikasikan sebelumnya. Seperti yang terlihat pada kolom deskripsi, setiap kumpulan data (*dataset*) akan mengoleksi jenis data yang berbeda dengan frekuensi yang berbeda pula.

Sebagai contoh, Umass Smart yang dikeluarkan oleh salah satu laboratorium di University of Massachusetts Amherst [4]. Umass Smart menyajikan beberapa jenis data yang berbeda dengan detail yang berbeda pula. Sebagai contoh, data yang dikoleksi tiap satu menit adalah sepuluh digit *unix timestamp* dan nilai *watt* yang terukur oleh meteran. Karena penelitian yang dilakukan hanya di beberapa rumah, data yang dikoleksi tidak menyertakan nomor identifikasi pelanggan.

Tabel 2.1 Informasi dasar dari beberapa *dataset* penelitian *smart grid* **[5]**

Nama	**Deskripsi**	**Frekuensi**	**Durasi**
Customer Behavior Trials	Hasil pengukuran *smart meter*; Hasil survei sebelum dan sesudah percobaan	Setiap 30 menit	September 2009 – Januari 2011
Low Carbon London	Hasil pengukuran *smart meter*; Data harga listrik; hasil survei perabotan dan perilaku	Setiap 30 menit	Januari 2013 – Desember 2013
PecanStreet	Data konsumsi listrik perumahan; data pengecasan kendaraan listrik	Setiap satu menit	Mei 2005 – Mei 2017
Umass Smart	Data konsumsi listrik perumahan	Setiap satu menit	Satu hari
Ausgrid Residents	Data konsumsi umum; data konsumsi beban terkontrol; data keluaran sel surya	Setiap 30 menit	Juli 2010 – Juni 2013



Gambar 2.1 Model konseptual *smart grid* **[2]**

Contoh lainnya adalah Customer Behavior Trials (CBT) yang dikeluarkan oleh badan regulator di Irlandia, Commission for Regulation of Utilities (CRU) [6]. CBT menunjukkan tiga informasi pada hasil pengukuran meteran listrik. Informasi tersebut adalah nomor identifikasi meteran, lima digit kode (tiga digit kode hari dan dua digit kode waktu), dan konsumsi listrik selama interval 30 menit (dalam kWh).

Gambar 2.1 merupakan model *smart grid* yang diajukan oleh National Institute of Standards and Technology (NIST), laboratorium sains di bawah naungan Departemen Perdagangan Amerika Serikat. NIST membagi *smart grid* menjadi tujuh domain untuk mendukung perencanaan, pengembangan kebutuhan, dokumentasi, dan penyusunan dari jaringan serta peralatan yang akan menyusun *smart grid*. Setiap domain tersebut memiliki tujuan dan pelayanan dasar pada *smart grid*. Penjelasan dasar dari tiap domain terdapat pada **Tabel 2.2**.

Untuk negara Indonesia, pengembangan *smart grid* sudah digencarkan oleh PLN (Perusahaan Listrik Negara) dalam bentuk pengenalan teknologi dan penyediaan suplai listrik secara nasional [7]. Meteran listrik dari PLN untuk pelanggan prabayar yang sudah memiliki layar digital dan dapat menampilkan beberapa informasi pelanggan merupakan salah satu teknologi yang dapat dikembangkan untuk menjadi penunjang *smart grid* [8].

Tabel 2.2 Domain dan tujuan/pelayanan pada model konseptual *smart grid* **[2]**

	Domain	**Tujuan/Pelayanan**
1	Pelanggan	Pengguna terakhir listrik. Bisa juga membuat, menyimpan, dan mengatur penggunaan energi. Secara sederhana, ada tiga tipe pelanggan yang didefinisikan dengan domain nya masing-masing, yaitu: perumahan, komersial, dan industri
2	Pasar	Operator dan peserta di pasar listrik
3	Penyedia Layanan	Organisasi yang menyediakan layanan kepada pelanggan listrik
4	Operasional	Manajer dari pergerakan listrik
5	Pembangkit	Penghasil listrik. Dapat juga menyimpan energi untuk distribusi yang akan datang.
6	Transmisi	Pembawa listrik dalam jumlah besar untuk jarak jauh. Dapat juga menyimpan dan menghasilkan listrik.
7	Distribusi	Distributor listrik untuk dan dari pelanggan. Dapat juga menyimpan dan menghasilkan listrik.

2.1.1 *Smart Meter*

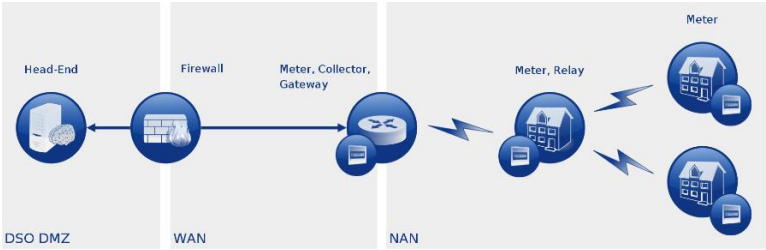
Smart meter merupakan alat yang dapat merekam statistik penggunaan dan melaporkan detailnya kepada perusahan utilitas, konsumen, dan pihak ketiga penyedia jasa [9]. Alat ini menjadi salah satu teknologi untuk mendukung kestabilan, ketahanan, dan keamanan dari *smart grid*. Tergantung dari infrastruktur pendukungnya, *smart meter* juga dapat digunakan untuk fungsi administratif lainnya, seperti notifikasi listrik mati dan penghentian pelayanan dari jarak jauh (*remote*).

Gambar 2.2 Contoh *smart meter* di Amerika Serikat **[10]**

Smart meter hadir sebagai *advanced meter reading* (AMR) dan juga *advanced metering infrastructure* (AMI) [11]. AMR biasanya merepresentasikan komunikasi satu arah yang memungkinkan perusahaan penyedia listrik untuk mendapatkan data meteran dari jarak jauh, sehingga dapat menghilangkan atau mengurangi kebutuhan mereka untuk melakukan pembacaan meteran secara manual. Biaya yang dikaitkan dengan AMR sebanding dengan biaya dari pengadaan *smart meter* pada umumnya. Tujuannya adalah untuk mendapatkan data meteran yang akurat sehingga penagihan yang tepat kepada pelanggan dapat tercapai. Namun, AMI memberikan fungsionalitas tambahan kepada perusahaan penyedia listrik untuk dapat diperluas hingga ranah pelanggan yang sebenarnya. AMI biasanya merepresentasikan komunikasi dua arah yang memungkinkan perusahaan penyedia listrik melakukan hal yang lebih dari sekadar mengoleksi pembacaan meteran dari jarak jauh. Biaya implementasi meteran dengan komunikasi dua arah jelas melampaui biaya yang hanya dengan komunikasi satu arah. Alasan dibalik hal tersebut adalah kebutuhan untuk menyediakan kapabilitas bagi pusat pengaturan untuk memiliki akses dan mengatur setiap meteran yang terdapat pada arsistektur jaringan. Pada saat yang bersamaan, pusat pengaturan juga memastikan meteran dapat memiliki kemampuan untuk melaporkan informasi kembali. Selain itu, juga terdapat kebutuhan untuk mengolah informasi tersebut sehingga dapat digunakan pada penagihan dan aplikasi lainnya yang membuat meteran menjadi "pintar". Aplikasi tersebut digunakan pada implementasi program *smart grid* yang memungkinkan perusahaan penyedia listrik mendapatkan manfaat.

2.1.2 Arsitektur Komunikasi *Smart Grid*

Advanced Metering Infrastructure (AMI) merupakan arsitektur yang menyediakan komunikasi dua arah untuk melaksanakan fungsi pengukuran tingkat lanjut [11]. Sistem ini berdasarkan arsitektur komunikasi pengukuran yang berstandar melalui ANSI (*American National Standards Institute*) C12.18, C12.19, C12.21, dan C12.22 [11]. ANSI C12.19 mendeskripsikan struktur tabel dari data aplikasi perusahaan penyedia listrik yang dilewatkan antara meteran listrik dan modul komunikasi meteran. ANSI C12.18 mendeskripsikan komunikasi tabel pada ANSI C12.19 melalui kabel optik. Lalu, ANSI C12.21 mendeskripsikan komunikasi tabel pada ANSI C12.19 melalui modem. Terakhir, ANSI C12.22 mendeskripsikan komunikasi tabel pada ANSI C12.19 melalui sebuah jaringan [12]. Setiap standar ANSI tersebut juga



Gambar 2.3 Komponen dan jaringan dari AMI **[13]**

diterbitkan melalui IEEE dengan kode yang berbeda, seperti ANSI C12.22 diterbitkan pada IEEE Std 1703-2012.

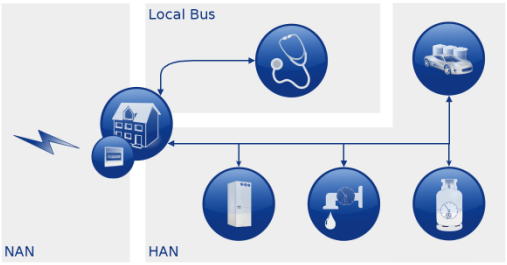
Pada sebuah jaringan *smart grid*, AMI biasanya tersusun atas beberapa jaringan dan memiliki komnponen utama. **Gambar 2.3** memberikan ringkasan dari seluruh komponen dan jaringan penyusun pada AMI, yaitu meteran, kolektor, dan *distribution system operator* (DSO) atau sisi perusahaan penyedia listrik.

Head-end system (HES) atau disebut sebagai sistem pengatur terletak pada jaringan perusahaan listrik. Pada umumnya, perusahaan listrik yang berperan sebagai operator sistem distribusi atau DSO. HES terhubung langsung dengan meteran, sehingga HES perlu diletakkan pada tempat khusus seperti zona demiliterisasi (DMZ). DMZ pada kasus ini memiliki arti bahwa jaringan DSO yang bersifat lokal akan terhubung dengan jaringan yang lebih besar, seperti internet.

Selain itu, juga terdapat infrastruktur pendukung lainnya pada DSO atau bagian perusahaan listrik. Sebagai contoh, data yang telah dikumpulkan akan diatur oleh *metering data management system* (MDMS) atau sistem manajemen data meteran. MDMS juga akan memetakan data untuk pelanggan yang relevan. Tergantung dari tingkat otomasi yang ditetapkan, data meteran akan memengaruhi aksi yang dilakukan DSO untuk menyeimbangkan jaringan listrik.

Kolektor atau disebut dengan konsentrator merupakan *gateway* yang berfungsi sebagai titik komunikasi untuk HES. Tergantung dari infrastruktur yang dibuat, kolektor dapat berupa meteran itu sendiri. Fungsi utamanya adalah sebagai perantara HES dengan meteran, kolektor lainnya, atau keduanya sekaligus dalam satu lingkungan atau *neighbourhood area network* (NAN).

Meteran atau *smart meter* terpasang pada bangunan milik pelanggan. Ketika sudah terintegrasi dengan kolektor, perangkat akan



Gambar 2.4 Skema HAN, NAN, dan *local bus* **[13]**

berkomunikasi secara langsung dengan HES. Sebagai sebuah meteran, perangkat dapat berkomunikasi dengan kolektor atau dapat berperan sebagai *relay* untuk meneruskan paket antara meteran lain dengan kolektor. Beberapa meteran juga menyediakan antarmuka untuk peralatan rumah tangga. Pada bagian konsumen, jaringan tersebut disebut dengan *home area network* (HAN).

Seperti yang telah disebutkan sebelumnya, AMI memiliki beberapa jaringan yang bergantung pada komponen dan protokol yang berbeda. Secara keseluruhan, terdapat tiga jaringan yang sering dideskripsikan, yaitu WAN, NAN, dan HAN.

WAN menghubungkan meteran atau kolektor dengan HES. WAN juga sering disebut sebagai jaringan *backhaul* atau pengalur. Komunikasi pada WAN biasanya berbasis *internet protocol* (IP) dan media teknologi informasi standar lainnya seperti kabel optik, DSL (*digital subscriber line*), atau GPRS (*general packet radio service*). Lalu, NAN menghubungkan meteran dan kolektor. Perangkat yang biasanya terdapat pada NAN adalah beberapa jenis meteran, seperti meteran listrik, gas, dan air. Terakhir, HAN berfungsi untuk mengintegrasikan tambahan meteran yang terdapat dalam satu bangunan. HAN memungkinkan adanya otomasi bangunan secara pintar. Contohnya adalah pengaturan otomatis dari *heating, ventilation, air conditioner* (HVAC) pada waktu puncak pemakaian listrik untuk menyeimbangkan jaringan listrik [13]. Dalam satu lingkungan, juga terdapat komponen lainnya seperti *local bus* yang berfungsi sebagai antarmuka untuk memeriksa ketahanan jaringan dalam skala kecil.

2.1.3 Keamanan *Smart Grid*

Pemikiran yang muncul pada sebagian besar masyarakat ketika mendengar istilah “teknologi baru,” “interkonektivitas,” “*data sharing*”,

dan “rekan bisnis” adalah adanya manfaat serta fungsionalitas baru yang terkandung di dalamnya [9]. Di sisi lain, ahli keamanan akan memikirkan tentang adanya risiko dibalik manfaat dan fungsionalitas yang ada. Ahli keamanan biasanya memiliki prinsip mengurangi hak istimewa dan membatasi akses untuk mengamankan data dan sumber daya. Oleh karenanya, divisi keamanan sering bertentangan dengan unit bisnis dan pengaturan keamanan juga dapat bertentangan dengan fungsionalitas yang baru. Seharusnya, pengaturan tersebut harus diterapkan dengan baik dan tidak mengganggu fungsionalitas yang ada. Tujuan dari pengaturan keamanan sejatinya adalah membantu fungsionalitas untuk beroperasi dengan benar dan melindunginya dari penyalahgunaan. Secara ideal, ahli keamanan dan unit bisnis akan bekerja sama untuk memastikan fungionalitas yang baru beroperasi secara aman, dengan berusaha untuk mempertahankan tujuan awal yang diinginkan.

Salah satu tujuan yang sering disebutkan dari *smart grid* adalah peningkatan keamanan dari jaringan listrik. Hal tersebut seakan menjadikan keamanan sebagai fitur tambahan. Namun, keamanan tersebut akan diperlukan untuk terintegrasi pada *smart grid* untuk bisa bekerja secara efektif. Kerahasiaan, integritas, dan ketersediaan merupakan prinsip utama dari keamanan informasi yang harus diaplikasikan untuk menjamin tercapainya tujuan dari *smart grid*.

Analisis data mempunyai peran penting pada *smart grid* dan akurasi atau integritas data menjadi krusial. Pengaturan keamanan yang sesuai diperlukan untuk menjamin data yang dikumpulkan tidak dimanipulasi. Sebagai contoh, *smart meter* akan mengirimkan data konsumsi listrik ke perusahaan penyedia listrik untuk keperluan tagihan dan operasional. Mekanisme keamanan seperti *hashing* dapat digunakan pada contoh tersebut sehingga perusahaan dapat mengesahkan data konsumsi dan memastikan pelanggannya tertagih dengan benar.

Smart grid dikatakan berhasil ketika manfaat yang didapatkan tidak menaikkan biaya secara signifikan bagi perusahaan penyedia listrik, dan khususnya bagi pelanggan. Implementasi dan biaya operasional *smart grid* harus tidak berpengaruh pada ketersediaan. Meskipun nilainya tidak diketahui secara pasti, PLN sebagai perusahaan penyedia listrik di Indonesia diperkirakan mengalami kerugian sekitar 10 triliun rupiah dalam setahun akibat pencurian listrik [14]. Penggelapan meteran menjadi salah satu metode yang banyak digunakan untuk pencurian listrik pada jaringan listrik saat ini. Hal tersebut diperkirakan juga terjadi dengan *smart meter* yang digunakan pada *smart grid*. Pengaturan keamanan yang

sesuai akan dibutuhkan untuk menjamin integritas *smart meter* serta tiap komponen pada *smart grid*. Meskipun menghilangkan kecurangan dan pencurian listrik adalah hal yang tidak mungkin, tetapi dengan mengurangi jumlah kejadiannya dapat menghasilkan penghematan yang signifikan bagi perusahaan. Akibatnya, perusahaan penyedia listik juga dapat menagih pelanggan dengan biaya yang lebih murah.

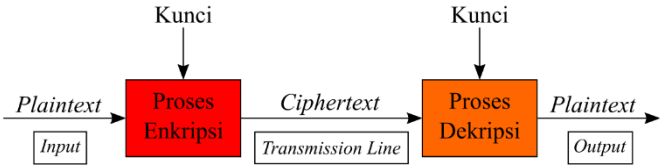
2.2 Kriptografi

Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan tersebut dikirim dari suatu tempat ke tempat lain [15]. Kriptografi berasal dari dua kata dalam bahasa Yunani, yaitu *kryptós* yang berarti "rahasia" dan *graphein* yang berarti "menulis".

Dalam kriptografi, ada beberapa komponen dasar yang membangunnya (**Gambar 2.5**), yaitu:

a. ***Plaintext*** (teks-biasa) adalah pesan atau informasi asli yang ingin disampaikan di antara dua pihak.
b. ***Ciphertext*** (teks-kode) adalah pesan atau informasi yang telah melalui proses enkripsi. *Ciphertext* berupa data yang terlihat acak atau tidak bermakna.
c. **Kunci** adalah sebuah kode yang terlibat dalam proses enkripsi dan dekripsi. Kunci yang digunakan pada kedua proses bisa sama atau berbeda tergantung algoritma yang digunakan.
d. **Enkripsi** adalah sebuah proses untuk mengubah *plaintext* menjadi *ciphertext* melalui suatu algoritma tertentu. Proses ini dilakukan oleh pihak pengirim.
e. **Dekripsi** adalah sebuah proses untuk mengubah *ciphertext* menjadi *plaintext* kembali melalui suatu algoritma tertentu. Proses ini dilakukan oleh pihak penerima.

Selain kelima komponen tersebut, ada istilah lain yang terdapat pada kriptografi, seperti ***cryptanalysis*** yang berupa sebuah analisis untuk bisa mengetahui informasi asli tanpa mengetahui kunci yang sebenarnya. Proses ini berguna untuk mengetahui kelemahan algoritma yang dapat digunakan oleh pihak ketiga untuk membobol informasi rahasia yang dikirimkan. Salah satu serangan yang dapat dilakukan untuk membobol suatu sistem kriptografi adalah ***brute-force***, yaitu dengan mencoba semua kunci yang mungkin untuk suatu *ciphertext* hingga menghasilkan *plaintext* yang dapat terbaca.



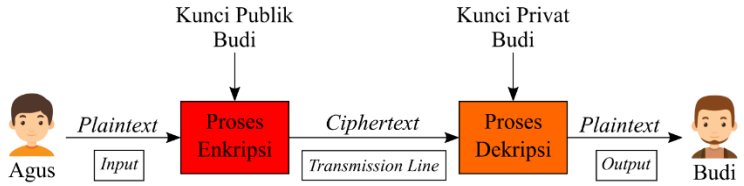
Gambar 2.5 Model sederhana kriptografi

2.2.1 Kriptografi Asimetris

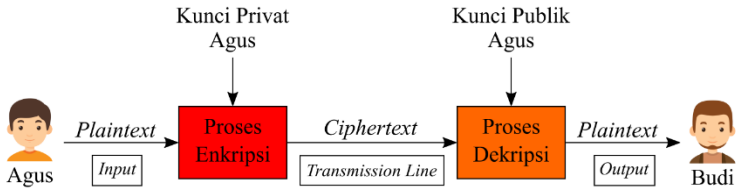
Kriptografi asimetris adalah salah satu jenis algoritma kriptografi yang sering juga disebut dengan algoritma kunci-publik [15]. Artinya, kunci yang digunakan untuk melakukan enkripsi dan dekripsi berbeda. Pada algoritma ini, kunci yang digunakan terbagi menjadi dua bagian, yaitu kunci umum (*public key*) dan kunci rahasia/pribadi (*private key*). Kunci umum adalah kunci yang boleh semua orang tahu (dipublikasikan), sedangkan kunci pribadi adalah kunci yang dirahasiakan atau hanya boleh diketahui oleh satu orang.

Kedua kunci tersebut berhubungan satu sama lain. Dengan kunci publik, seseorang dapat mengenkripsi pesan, tetapi dia tidak dapat mendekripsinya. Hanya orang yang memiliki kunci pribadi yang dapat mendekripsi pesan tersebut. Seperti yang terlihat pada **Gambar 2.6**, ketika Agus ingin mengirimkan pesan kepada Budi, Agus akan menggunakan kunci publik milik Budi untuk mengenkripsi pesan tersebut. Lalu, Budi yang akan menggunakan kunci privatnya untuk mendekripsi pesan tersebut.

Selain untuk proses pengamanan, kriptografi asimetris juga dapat berfungsi sebagai tanda tangan digital. Tanda tangan digital ini tidak memberikan kerahasiaan seperti proses sebelumnya. Layaknya tanda tangan pada umumnya, proses ini hanya berguna untuk menandai pesan sehingga penerima tahu bahwa pesan tersebut dikirim oleh pengirim yang bersangkutan.



Gambar 2.6 Proses pengamanan menggunakan kriptografi asimetris



Gambar 2.7 Proses tanda tangan digital menggunakan kriptografi asimetris

Selain untuk proses pengamanan, kriptografi asimetris juga dapat berfungsi sebagai tanda tangan digital. Tanda tangan digital ini tidak memberikan kerahasiaan seperti proses sebelumnya. Layaknya tanda tangan pada umumnya, proses ini hanya berguna untuk menandai pesan sehingga penerima tahu bahwa pesan tersebut dikirim oleh pengirim yang bersangkutan.

Seperti yang terlihat pada **Gambar 2.7**, proses enkripsi tidak lagi menggunakan kunci milik Budi, tetapi menggunakan kunci privat milik Agus. Budi dan juga orang lainnya dapat dengan membuka pesan yang dikirimkan oleh Agus dengan menggunakan kunci publik miliknya. Proses tersebut membuktikan bahwa Agus adalah pengirim yang sebenarnya dari pesan tersebut. Pihak ketiga harus mengetahui kunci privat Agus jika ingin memalsukan pesan yang dikirimkan Agus.

Kegunaan lainnya dari kriptografi asimetris adalah untuk pertukaran kunci. Proses ini dapat dilakukan oleh dua pihak yang menginginkan kunci untuk penggunaan kriptografi simetris. Berbeda dengan asimetris, kriptografi simetris menggunakan kunci yang sama pada pengirim dan penerima. Dengan proses pertukaran kunci, pengirim dan penerima masing-masing akan membuat informasi secara privat dan mengirimkan informasi publik ke pihak lawan. Pihak ketiga tidak dapat mendapatkan informasi yang penting dari informasi publik tersebut. Namun, pengirim dan penerima dapat menggunakan informasi publik tersebut untuk mendapatkan kunci simetris yang diinginkan.

Beberapa jenis kriptografi asimetris memiliki kegunaannya masing-masing. Sebagai contoh, *elliptic-curve cryptography* (ECC) atau kriptografi kurva elips dan algoritma RSA dapat digunakan untuk ketiga proses yang telah disebutkan. Selain itu, juga terdapat algoritma Diffie-Hellman yang hanya untuk proses pertukaran kunci dan *Digital Signature Algorithm* (DSA) yang hanya untuk proses tanda tangan digital.

Tabel 2.3 Aplikasi sistem kriptografi kunci-publik/asimetris **[16]**

Algoritma	Kerahasiaan Data	Tanda Tangan Digital	Pertukaran Kunci
RSA	Ya	Ya	Ya
ECC	Ya	Ya	Ya
Diffie-Hellman	Tidak	Tidak	Ya
DSA	Tidak	Ya	Tidak

2.2.2 RSA

Pada tahun 1977, Ron Rivest, Adi Shamir, dan Len Adleman dari MIT (Massachusetts Institute of Technology) mengembangkan sebuah algoritma sebagai jawaban dari tantangan untuk membuat sistem kriptografi kunci publik yang aman [16]. Mereka berhasil menemukan algoritma yang menurut mereka sulit dipecahkan tetapi perhitungan yang digunakan tetap sederhana [17]. Atas pencapaian yang telah mereka lakukan, algoritma tersebut dinamakan sesuai dengan singkatan nama belakang mereka, yaitu Rivest-Shamir-Adleman atau RSA.

Skema yang digunakan pada RSA adalah dengan memodelkan *plaintext* dan *ciphertext* menjadi sebuah bilangan bulat antara nol dan n – 1 untuk nilai n. Nilai dari n biasanya sebesar 1024 bit atau 309 digit desimal. Oleh karenanya, n akan bernilai kurang dari 2^{1024}.

RSA memanfaatkan prinsip dari eksponensial. *Plaintext* dienkripsi dalam blok-blok yang mempunyai ukuran dengan nilai biner kurang dari bilangan tertentu n. Oleh karena itu, ukuran dari blok data akan sebesar i bit dengan $2^i < n \leq 2^{i+1}$. Proses enkripsi dan dekripsi dilakukan dengan proses berikut ini:

$$C = M^e \bmod (n).$$
$$M = C^d \bmod (n) = (M^e)^d \bmod (n) = M^{ed} \bmod (n).$$

C adalah blok data *ciphertext* dan M adalah blok data *plaintext*. Pengirim dan penerima mengetahui nilai dari n dan e. Namun, nilai d hanya diketahui oleh penerima. Oleh karena itu, RSA ini berupa algoritma enkripsi asimetris atau kunci-publik dengan kunci umum $PU = [e, n]$ dan kunci privat $PR = [d, n]$.

Algoritma RSA didasarkan pada beberapa teori matematika seperti Teorema Euler-Fermat yang menyatakan bahwa untuk nilai a dan n yang **relatif prima** atau faktor pembagi terbesar (FPB) dari a dan n bernilai 1, maka:

$$a^{\phi(n)} \equiv 1 \left(\bmod (n)\right) \quad \textbf{(1)}$$

$\Phi(n)$ merupakan *Euler's totient function* yang menghasilkan angka jumlah dari bilangan bulat positif kurang dari *n* yang relatif prima terhadap *n*. Sebagai contoh, $\Phi(9) = 6$ karena angka 9 relatif prima dengan 1, 2, 4, 5, 7, dan 8. Secara umum, rumus dari fungsi tersebut adalah sebagai berikut:

$$\phi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \ldots \left(1 - \frac{1}{p_r}\right) \quad \textbf{(2)}$$

P_1, p_2, ... , p_r adalah faktor-faktor prima dari *n*.

Untuk *n* yang berupa bilangan prima, nilai dari $\phi(n) = n - 1$. Jika *n* merupakan perkalian antara dua bilangan prima *p* dan *q*, maka sesuai dengan sifat dasar *totient function*, nilai $\phi(n) = \phi(p) \cdot \phi(q)$ atau:

$$\phi(n) = (p - 1) \cdot (q - 1). \quad \textbf{(3)}$$

Nilai dari *e* dipilih sehingga bersifat relatif prima dengan $\Phi(n)$ pada persamaan **(3)**. Sedangkan nilai dari *d* dihitung sehingga memenuhi persamaan berikut:

$$e \cdot d \equiv 1 \left(\mathrm{mod}\left(\phi(n)\right)\right) \quad \textbf{(4a)}$$

$$d \equiv e^{-1} \left(\mathrm{mod}\left(\phi(n)\right)\right) \quad \textbf{(4b)}$$

Nilai pangkat -1 pada e^{-1} pada persamaan **(4b)** menandakan bahwa *d* merupakan pengali invers dari *e* pada ***ring*** bilangan modulo $\Phi(n)$. Pengali invers yang dimaksud adalah pasangan dari suatu bilangan sehingga hasil kali keduanya ekuivalen dengan satu pada modulo bilangan tertentu, seperti pada persamaan **(4a)**.

Dengan beberapa teori matematika tersebut, RSA bisa bekerja dengan sempurna. Proses enkripsi dan dekripsi dari RSA seperti pada penjelasan sebelumnya dapat ditulis ulang menjadi:

$$M^{e \cdot d} \equiv M^{k \cdot \phi(n)+1} \left(\mathrm{mod}\,(n)\right) \quad (\text{untuk nilai } k \text{ tertentu}) \quad \textbf{(5)}$$

Untuk semua nilai *M* yang tidak habis dibagi oleh *p*, persamaan **(1)** menjadi $M^{p-1} \equiv 1 \left(\mathrm{mod}\,(p)\right)$, dan karena (p – 1) membagi $\Phi(n)$, persamaan tersebut menjadi $M^{k \cdot \phi(n)+1} \equiv M \left(\mathrm{mod}\,(p)\right)$. Hasil persamaan akhir yang didapatkan ini juga berlaku untuk nilai *q* sehingga menjadi $M^{k \cdot \phi(n)+1} \equiv M \left(\mathrm{mod}\,(q)\right)$. Jika digabungkan, persamaan **(5)** akan menyatakan bahwa untuk semua nilai *M*,

$$M^{e \cdot d} \equiv M^{k \cdot \phi(n)+1} \equiv M \left(\mathrm{mod}\,(n)\right) \quad (0 \leq M < n) \quad \textbf{(6)}$$

Persamaan **(6)** menjadi bukti akhir bahwa nilai *M* yang telah melewati proses enkripsi dan dekripsi pada RSA ($M^{e \cdot d}$) dapat dikembalikan menjadi seperti semula (*M*).

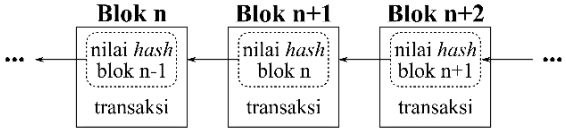
Secara ringkas, berikut ini adalah langkah-langkah yang perlu dilakukan dalam RSA:

- Bilangan prima *p* dan *q* dipilih dengan $p \neq q$
- *n* dihitung dengan cara $n = p \times q$
- $\Phi(n)$ dihitung dengan $\phi(n) = (p-1)(q-1)$
- e dipilih sehingga memenuhi FPB $(\phi(n), e) = 1$ dan $1 < e < \phi(n)$
- d dicari sehingga memenuhi persamaan $(\mathbf{3a})$ $e \cdot d \equiv 1 \left(\text{mod } (\phi(n))\right)$ atau $(\mathbf{3b})$ $d \equiv e^{-1} \left(\text{mod } (\phi(n))\right)$
- Kunci publik *PU* adalah [*e*, *n*]
- Kunci privat *PR* adalah [*d*, *n*]
- Proses enkripsi : $C = M^e \text{ mod } (n)$, dengan syarat $M < n$
- Proses dekripsi : $M = C^d \text{ mod } (n)$

Salah satu variabel penting yang terdapat pada RSA adalah pemilihan ukuran kunci yang digunakan. Ukuran kunci yang besar akan memberikan keamanan lebih baik, tetapi sistem akan berjalan semakin lambat [16]. Sementara itu, ukuran kunci yang terlalu kecil akan memudahkan sistem untuk bisa dibobol oleh pihak yang tidak bertanggung jawab. Oleh karena itu, NIST memberikan rekomendasi untuk tidak menggunakan kunci yang berukuran 1024 bit atau yang lebih kecil untuk pengamanan informasi penting di pemerintahan [18].

2.3 *Blockchain*

Blokchain adalah sebuah rangkaian blok data yang berisikan informasi unik dan dilengkapi dengan kode *hash* sebagai bukti keterkaitan tiap blok yang dihasilkan. *Blockchain* berhasil dipopulerkan oleh sebuah mata uang digital atau *cryptocurrency* yang bernama Bitcoin. Meskipun bermula dari permasalahan pada Bitcoin, *blockchain* dapat berdiri sendiri tanpa penggunaan *cryptocurrency* [19].



Gambar 2.8 Skema *blockchain* sederhana

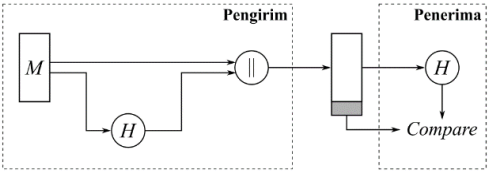
Blockchain dapat diasumsikan sebagai arsip transaksi yang dikumpulkan pada blok-blok dengan penanda waktu atau ***timestamp***. Setiap blok teridentifikasi dengan suatu nilai *hash*. Namun, setiap blok tersebut mereferensikan nilai *hash* dari blok yang ada sebelumnya. Oleh karena itu, ikatan antar blok akan terjadi dan membuat sebuah rantai blok atau *blockchain* seperti pada **Gambar 2.8**. Skema tersebut akan berulang untuk blok-blok selanjutnya, tetapi pengecualian terjadi pada blok pertama dari rantai tersebut, yang dinamakan *genesis*. Blok *genesis* ini akan menjadi penanda secara umum pada seluruh jaringan *blockchain*, dan tanpa memiliki blok pendahulu.

Beberapa karakteristik penting yang dimiliki oleh jaringan *blockchain* adalah sebagai berikut [20]:

a. **Konsensus**: Sebuah transaksi dapat dinyatakan valid ketika semua partisipan setuju dengan validitasnya.
b. **Asal**: Partisipan mengetahui sumber datangnya aset dan perubahan kepemilikannya sepanjang waktu.
c. **Kekekalan**: Tidak ada partisipan yang dapat merubah sebuah transaksi setelah tercatat di penyimpanan utama atau ***ledger***. Jika sebuah transakasi sedang terjadi kesalahan, transaksi baru harus digunakan untuk membalikkan kesalahannya, dan kedua transaksi akan terlihat.
d. **Penutup**: Sebuah *ledger* umum dapat menjadi satu tempat tujuan untuk menentukan kepemilikan sebuah aset atau penyelesaian sebuah transaksi.

2.3.1 Fungsi *Hash*

Fungsi *hash* merupakan suatu fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya ke dalam urutan biner dengan panjang yang tetap. Fungsi ini digunakan pada berbagai aplikasi pengamanan dan protokol internet. Beberapa contoh kegunaannya adalah untuk autentikasi pesan, tanda tangan digital, dan penyimpanan *password* [16].

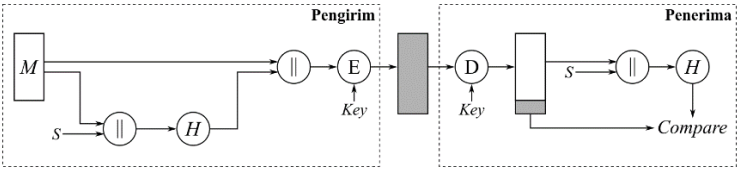


Gambar 2.9 Skema autentikasi sederhana

Autentikasi pesan berguna untuk mengecek integritas dari sebuah pesan. Dengan kata lain, proses ini memastikan data yang diterima sama persis dengan yang dikirim atau tanpa adanya modifikasi data. Langkah pertama yang perlu dilakukan untuk proses ini adalah pengirim menghitung nilai *hash* sebagai keluaran fungsi dari pesan yang akan dikirim dan mengirimkan pesan beserta nilai *hash*-nya. Selanjutnya, penerima hanya perlu mencocokkan pesan dan nilai *hash* yang diterima dengan menghitung nilai *hash* dari pesan tersebut. Nilai *hash* dari pengirim dan hasil dari proses perhitungan akan bernilai sama jika pesan yang terkirim tidak mengalami perubahan dalam proses pengiriman. Namun, proses ini tidak memberikan kerahasiaan pada data yang dikirimkan. Selain itu, serangan *man-in-the-middle* bisa terjadi dengan cara pelaku mengambil alih data asli dari pengirim dan mengirim pesan palsu dan nilai *hash*-nya kepada penerima.

Oleh sebab itu, penambahan proses enkripsi dilakukan dalam proses autentikasi sehingga komunikasi berlangsung secara aman. Proses enkripsi ini diletakkan di antara langkah-langkah autentikasi. Contoh pertama adalah enkripsi yang diletakkan setelah proses menghitung nilai *hash* untuk implementasi yang tidak memerlukan kerahasiaan data secara menyeluruh. Contoh berikutnya adalah enkripsi yang diletakkan setelah proses penggabungan pesan dengan nilai *hash* termodifikasi seperti pada **Gambar 2.10**.

Salah satu bagian yang perlu diperhatikan pada skema autentikasi pada **Gambar 2.10** adalah penambahan kode *S* sebelum perhitungan nilai *hash*. Proses tersebut merupakan langkah tambahan untuk mengantisipasi pihak ketiga yang dapat membuat pesan palsu. Kode *S* tersebut disepakati oleh pihak pengirim dan penerima sebelum komunikasi berlangsung. Oleh karena itu, pihak ketiga yang mencoba mengirim pesan palsu kepada penerima dapat terdeteksi meskipun kunci enkripsi berhasil diketahui sebelumnya.



Gambar 2.10 Contoh skema autentikasi dengan enkripsi

Dalam beberapa tahun terakhir, fungsi *hash* yang paling sering digunakan adalah Secure Hash Algorithm (SHA). Hal itu dikarenakan kelemahan dasar dari fungsi *hash* lainnya telah ditemukan melalui *cryptanalysis*. SHA sendiri menjadi salah satu dari beberapa fungsi *hash* yang terstandarisasi. SHA dikembangkan oleh NIST dan dipublikasikan sebagai standar pengolahan informasi di tahun 1993. Setelah beberapa kelemahan ditemukan pada **SHA-0** yang dirilis tersebut, NIST merilis revisi dari SHA di tahun 1995 dan dinamakan **SHA-1**. Hingga saat ini, ada dua versi SHA yang masih belum terbobol secara praktis, yaitu **SHA-2** dan **SHA-3**. Meski begitu, SHA-2 yang dirilis tahun 2001 saat ini telah menunjukkan pelemahan dalam proteksi data [21].

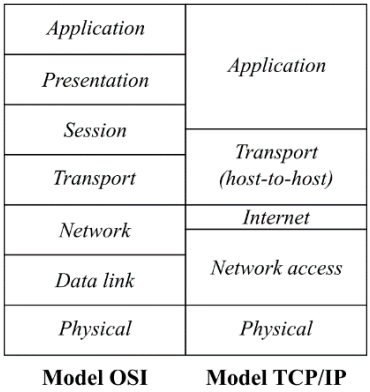
2.4 Arsitektur Protokol

Ketika komputer dan/atau alat pengolahan data lainnya saling tukar menukar data, prosedur yang terlibat dapat menjadi rumit. Sebagai contoh, transfer berkas antara dua komputer harus memerlukan jalur data antara keduanya, seperti melalui jaringan komunikasi. Namun, banyak hal lainnya yang diperlukan seperti cara menyepakati jalur data yang digunakan dan fungsi penerjemah format berkas yang berbeda [22].

Oleh karena itu, harus ada kesepakatan tingkat tinggi yang terdapat di antara dua sistem komputer. Tugas-tugas yang diperlukan akan dipecah menjadi beberapa modul yang diimplementasikan secara terpisah. Kumpulan modul tersebut yang disusun secara vertikal terdapat pada arsitektur protokol. Setiap lapisan (*layer*) akan memiliki fungsinya tersendiri yang dibutuhkan untuk komunikasi dengan sistem lainnya. Semakin ke bawah lapisan pada suatu perangkat, fungsi yang dijalankan akan semakin primitif.

Ketika dua perangkat berkomunikasi, lapisan fungsi yang sama harus terdapat pada keduanya. Komunikasi terjadi ketika lapisan yang sama (*peer*) di kedua perangkat saling terhubung. Komunikasi tersebut berdampak pada blok data yang perlu diformat untuk mematuhi kumpulan aturan atau ketentuan yang dikenal dengan **protokol**.

Arsitektur protokol yang paling penting saat ini adalah **Transmission Control Protocol** dan **Internet Protocol** yang membentuk rangkaian protokol **TCP/IP**. Protokol ini yang menjadi dasar dari sistem Internet. Arsitektur lainnya yang cukup dikenal adalah model **Open Systems Interconnection (OSI)**. Model OSI merupakan arsitektur yang telah terstandar dan sering digunakan untuk menjelaskan fungsi-fungsi komunikasi, tetapi jarang diimplementasikan untuk saat ini.



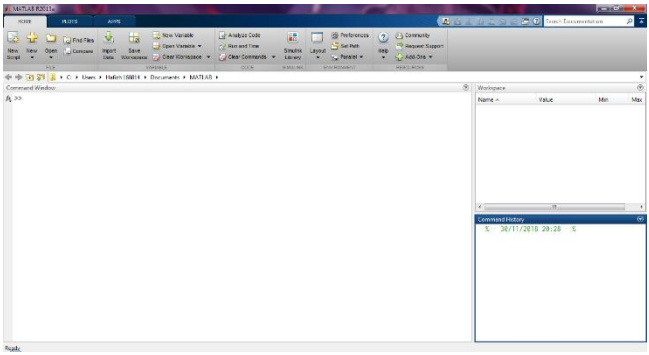
Gambar 2.11 Perbandingan *layer* pada model OSI dan model TCP/IP **[22]**

Penjelasan singkat dari setiap *layer* dalam TCP/IP adalah sebagai berikut:

- *Physical layer* menjelaskan tentang pengaturan koneksi fisik di antara perangakat transmisi seperti komputer dan jaringan. Salah satu contohnya adalah karakteristik medium transmisi.
- *Network access layer* menjelaskan tentang pengaturan dari pertukaran data pada komputer yang terkait pada jaringan. Salah satunya adalah perutean data antara dua sistem pada jaringan.
- *Internet layer* menjelaskan tentang prosedur yang dibutuhkan untuk bisa menghubungkan perangkat yang berada di jaringan yang berbeda.
- *Transport layer* menjelaskan tentang protokol yang dapat diandalkan melayani hubungan *end-to-end*.
- *Application layer* menjelaskan tentang logika yang dibutuhkan untuk mendukung penggunaan aplikasi pengguna yang beragam.

2.5 MATLAB

MATLAB merupakan sebuah lingkungan pemrograman yang dikhususkan untuk pengembangan algoritma, analisis data, visualisasi, dan perhitungan numerik. MATLAB juga merupakan sebuah bahasa pemrograman tersendiri yang dikembangkan oleh perusahaan bernama MathWorks [23].



Gambar 2.12 Cuplikan layar dari aplikasi MATLAB R2013a

Beberapa kegunaan dari MATLAB adalah:

- Komputasi matematis
- Pengembangan algoritma
- Pemodelan, simulasi, dan pembuatan purwarupa
- Analisis data dan visualisasi
- Penggambaran di bidang teknik dan sains

BAB 3
PERANCANGAN SISTEM

Setelah melakukan studi literatur untuk mendapatkan teori dasar yang digunakan pada penelitian ini, tahapan utama yang berikutnya dilakukan adalah merancang simulasi program yang akan dijalankan. Program tersebut akan dirancang untuk menggambarkan hanya satu sesi atau siklus kegiatan dalam jaringan *smart grid*, tetapi dijalankan secara berulang-ulang. Hal tersebut dikarenakan program yang dibuat akan menjadi salah satu komponen penting dalam penerapannya pada jaringan di masa yang akan datang.

3.1 Potongan Fungsi

Kode program dalam penelitian ini akan terbagi menjadi potongan fungsi. Fungsi yang dipotong-potong ini bertujuan untuk melihat performa awal suatu fungsi jika variabel di dalamnya diubah-ubah. Selain itu, program akan lebih mudah dianalisis karena setiap fungsi memiliki masukan dan keluaran tersendiri yang akan saling terkait satu sama lain. Dua fungsi utama yang akan dibahas dalam penelitian ini adalah fungsi RSA dan fungsi *hash*.

3.1.1 Fungsi RSA

Berikut ini adalah kode program dari fungsi RSA di aplikasi MATLAB yang dikembangkan oleh Andrew Janke [24]:

```
import javax.crypto.Cipher;
plaintext = 'foobar';

cipher = Cipher.getInstance('RSA');
keygen =
java.security.KeyPairGenerator.getInstance('RSA');
keyPair = keygen.genKeyPair();
cipher.init(Cipher.ENCRYPT_MODE, keyPair.getPrivate());
plaintextUnicodeVals = uint16(plaintext);
plaintextBytes = typecast(plaintextUnicodeVals, 'int8');
ciphertext = cipher.doFinal(plaintextBytes)'  %'

cipher.init(Cipher.DECRYPT_MODE, keyPair.getPublic());
decryptedBytes = cipher.doFinal(ciphertext);
decryptedText = char(typecast(decryptedBytes, 'uint16'))'
```

Kode program tersebut memiliki sebuah masukan berupa *string* dengan variabel `plaintext` dan menghasilkan keluaran berupa deretan angka dengan variabel `ciphertext`, atau sama dengan proses enkripsi. Selain itu, kode tersebut juga bisa ditambahkan dengan proses dekripsi untuk pengecekan dengan keluaran berupa variabel `decryptedText`. Dalam prosesnya, program ini memanfaatkan Java Virtual Machine (JVM) sehingga sebagian besar kode yang dibuat menggunakan bahasa pemrograman Java.

Cara kerja dari kode program fungsi RSA tersebut dapat dipecah menjadi beberapa proses seperti berikut:

a. Fungsi `import` pada awal kode berfungsi untuk memanggil *package* atau *class* dari JVM yang sudah terdapat pada aplikasi MATLAB. *Class* yang dipakai pada program adalah javax.crypto.Cipher. *Class* tersebut menyediakan fungsionalitas dari sandi kriptografi untuk proses enkripsi dan dekripsi.
b. *Class* `Cipher.getInstance('RSA')` berfungsi untuk mendefinisikan jenis sandi yang akan digunakan pada program. Dalam kasus ini, jenis sandi yang digunakan adalah RSA.
c. Pembuatan kunci pada program didefinisikan pada *class* `KeyPairGenerator.getInstance('RSA')` dan dibuat menggunakan `genKeyPair()`. Kunci yang dibuat juga dapat diatur ukuran bitnya menggunakan `initialize(1024)` dengan mengganti angka yang berada di dalam kurung.
d. RSA menggunakan angka sebagai masukan *plaintext* pada proses enkripsi. Oleh karena itu, *plaintext* yang telah didefinisikan diubah menjadi angka menggunakan format *16-bit unsigned integer* (uint16) dan diubah kembali menjadi format *8-bit signed integer* (int8) menggunakan fungsi `typecast()`.
e. Proses enkripsi atau dekripsi pada program diawali dengan `init(...)` dan diakhiri dengan `doFinal(...)`. Isi dari dalam kurung fungsi `init` adalah jenis proses yang dilakukan dan kunci yang akan digunakan. `ENCRYPT_MODE` menunjukkan proses enkripsi dan `getPrivate()` menunjukkan penggunaan kunci privat. Sedangkan isi dari dalam kurung fungsi `doFinal` adalah nama variabel yang akan diolah.

3.1.2 Fungsi *Hash*

Berikut ini adalah kode program sederhana fungsi *hash*:

```
data = 'kvIf7#KHAP0#20180827T020745Z#A#12345.67#1234.56#';
prev_hash =
'000202532D961167716C983CA38C72DAFE9CCAABFD9FF50B9B1B549DC
55F6543';
string = add_chara(prev_hash,data);

sha256hasher = System.Security.Cryptography.SHA256Managed;
sha256 = uint8(sha256hasher.ComputeHash(uint8(string)));

while sha256(1)~=0 || sha256(2)>=16
    string = add_chara(prev_hash,data);
    sha256 =
uint8(sha256hasher.ComputeHash(uint8(string)));
end

hash = reshape(dec2hex(sha256)',64,1)';
prev_hash = hash;
```

Kode program tersebut memiliki masukan berupa barisan karakter (*string*) yang didefinisikan dengan variabel `data` dan keluaran berupa barisan bilangan *hexadecimal* dengan variabel `HSH`. Karena menggunakan JVM dalam prosesnya, kode program tersebut juga menggunakan bahasa pemrograman Java.

Cara kerja dari kode program fungsi *hash* tersebut dapat dipecah menjadi beberapa proses seperti berikut:

a. *String* pada variabel `data` ditambahkan dengan karakter acak dan potongan nilai *hash* data sebelumnya menggunakan fungsi `add_chara(...)`. Hal ini ditujukan sebagai bukti kerja (*proof of work*) dari *server* yang bertugas untuk menjalankan *blockchain*.

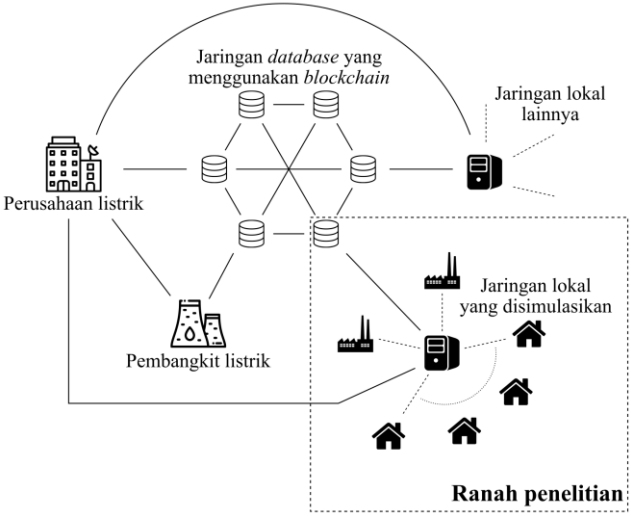
b. Proses perhitungan nilai *hash* sendiri dilakukan menggunakan JVM dengan memanggil *class* `System.Security.Cryptography.SHA256Managed` dan fungsi `ComputeHash(...)`. Jenis fungsi *hash* yang digunakan dalam penelitian ini adalah SHA-2 dengan keluaran 256 bit atau SHA-256. Isi dari dalam kurung fungsi `ComputeHash` adalah masukan yang ingin dihitung nilai *hash*-nya dalam format bilangan *uint8*. Keluaran yang dihasilkan juga perlu dikonversi ke format bilangan *uint8*.

c. *Proof of work* yang dimaksud pada bagian sebelumnya merupakan salah satu proses yang perlu dilakukan sebagai bukti bahwa pekerjaan untuk menghitung nilai *hash* telah dilakukan untuk sebuah masukan *string*. Pada kode program tersebut, *proof of work* didefinisikan dengan mencari nilai *hash* yang memiliki awalan nol untuk tiga digit pertama.
d. Setelah nilai *hash* yang memenuhi aturan yang ditetapkan berhasil ditemukan, *string* tersebut dinyatakan valid dan bisa menjadi blok baru dalam *blockchain*.

Proses yang dilakukan pada kode program fungsi *hash* ini hampir sama dengan cara pada transakasi Bitcoin, tetapi sistem pada Bitcoin mengharuskan nilai *hash* yang dihasilkan berada di bawah suatu nilai atau dikenal dengan istilah ***target***. Sebagai perbandingan, nilai *target* saat ini memiliki awalan nol untuk 18 digit pertama [25].

3.2 Desain Jaringan *Smart Grid*

Jaringan *smart grid* memiliki beberapa komponen penting yang terhubung di dalamnya. *Smart grid* juga dapat memiliki model yang berbeda-beda tergantung dari aplikasi, permasalahan, atau domain yang dibutuhkan [2]. Untuk penelitian ini, **Gambar 3.1** menggambarkan

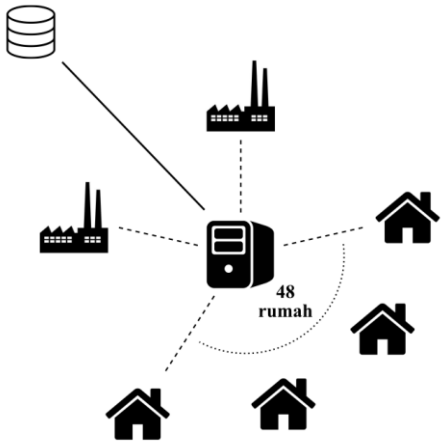


Gambar 3.1 Rancangan jaringan *smart grid* dengan *blockchain*

rancangan jaringan *smart grid* yang menggunakan tambahan sistem *blockchain*. Beberapa komponen penting yang terdapat pada gambar adalah: perusahaan listrik, pembangkit listrik, jaringan listrik lokal, dan jaringan *database* yang menggunakan *blockchain*. **Gambar 3.1** ini hanya memberikan gambaran dasar tentang komunikasi yang terjadi pada suatu jaringan *smart grid* secara menyeluruh. Jaringan yang terdapat pada gambar juga tidak merepresentasikan distribusi listrik di antara tiap komponen. Penelitian ini hanya berfokus terhadap satu jaringan lokal yang terhubung dengan *database*.

Dalam penelitian ini, jaringan *smart grid* yang disimulasikan akan terlihat seperti pada **Gambar 3.2**. Jaringan tersebut tersusun dari 48 rumah dan dua pabrik yang terhubung dengan *server* lokal. Sehingga, jumlah pengguna yang terdapat pada jaringan adalah 50 pengguna. Selain itu, *server* lokal terhubung dengan sebuah *database*. *Database* ini berperan dalam penyimpanan data yang menggunakan sistem *blockchain*.

Topologi jaringan yang digunakan pada penelitian ini adalah ***star topology*** atau topologi bintang. Topologi jenis ini dipilih karena penelitian ini diasumsikan hanya pada skala lokal. Untuk skala yang lebih besar, topologi yang dapat digunakan adalah *tree topology* atau topologi pohon. Topologi pohon memiliki nama lain dalam sistem distribusi tenaga, yaitu ***radial grid***. Topologi jenis ini menjadi salah satu topologi yang paling umum digunakan di Amerika Serikat [9].



Gambar 3.2 Desain jaringan lokal yang disimulasikan

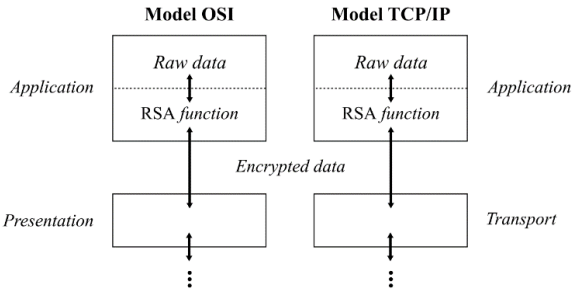
3.2.1 Peletakan Program dalam Jaringan

Program yang dibuat dalam penelitian ini akan menjadi salah satu komponen dalam jaringan *smart grid*. Jaringan tersebut bekerja layaknya jaringan telekomunikasi lainnya. Oleh karena itu, program akan dirancang untuk bisa kompatibel dengan desain jaringan telekomunikasi yang sudah ada. Program pertama yang berupa fungsi RSA diletakkan dekat dengan ***raw data*** atau data kasar. Lalu, program kedua yang berupa fungsi *hash* diletakkan sebelum proses penyimpanan oleh *server*.

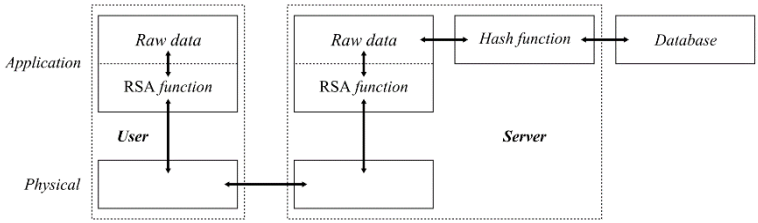
Pada model OSI, fungsi RSA direncanakan untuk beroperasi di antara *application layer* dan *presentation layer*. Hal tersebut dikarenakan masukan dan keluaran dari program hanya berupa barisan bit data. Bentuk tersebut merupakan bentuk paling sederhana dalam jaringan telekomunikasi. Barisan bit data informasi menjadi salah satu penyusun *frame* pada model OSI.

Pada model TCP/IP, fungsi RSA direncanakan untuk beroperasi di antara *application layer* dan *transport layer*. Sama dengan model OSI sebelumnya, hal tersebut dikarenakan masukan dan keluaran program hanya berupa barisan bit data. Salah satu hal yang menjadi pembeda dengan model OSI adalah bit data informasi menjadi salah satu penyusun TCP *segment* dengan adanya penambahan TCP *header*.

Jika fungsi RSA diletakkan pada setiap perangkat pada jaringan, fungsi *hash* hanya akan diletakkan pada *server* sebagai bagian dari pembuatan *blockchain*. Setelah *raw data* berhasil melewati proses dekripsi pada *server*, data tersebut akan diolah melalui fungsi *hash* untuk mendapatkan nilai *hash*-nya. Nilai *hash* tersebut akan menjadi bagian dari blok pada pembuatan *blockchain*. *Blockchain* ini akan menjadi catatan utama dari penggunaan listrik yang tersimpan pada suatu *database*.



Gambar 3.3 Peletakan fungsi RSA pada kedua model jaringan



Gambar 3.4 Skema lengkap dari penerapan program pada komunikasi dengan prinsip *end-to-end* antara *user* dan *server*

Pengamanan data atau *blockchain* di dalam *database* sudah berada di luar pembahasan dalam penelitian ini. Namun, hal tersebut nantinya akan menjadi kewenangan pihak yang telah disepakati, seperti pihak penyedia layanan listrik dan/atau pemerintah.

Di dalam jaringan yang akan diterapkan nantinya, terdapat pula penerapan dari prinsip ***end-to-end***. Prinsip *end-to-end* merupakan prinsip jaringan yang menempatkan *application layer* hanya pada titik akhir komunikasi. Titik perantara tidak memiliki hak untuk mengetahui data yang akan dilewatkan.

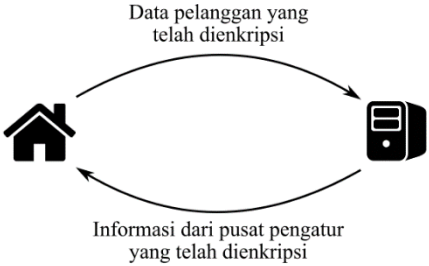
3.2.2 Proses Pertukaran Data

Pada penelitian ini, jaringan *smart grid* tidak menggunakan medium kabel untuk berkomunikasi atau lebih dikenal dengan istilah *wireless*. Jaringan akan memanfaatkan kanal frekuensi yang masih tersedia atau bekerjasama dengan operator telekomunikasi untuk menggunakan kanal yang sudah ada.

Pada perangkat *smart meter* yang dimiliki oleh pelanggan, modul komunikasi seperti antena dan pengolah data menjadi komponen yang digunakan untuk berkomunikasi dengan pusat pengatur data. Perangkat tersebut dapat bekerja layaknya sebuah *smartphone* pada jaringan *smart grid*.

Jika mengikuti standar yang telah ada, jaringan *smart grid* diharuskan memiliki sistem keamanannya sendiri. Namun, dalam penelitian ini, sistem keamanan yang baru diletakkan sebelum proses pengamanan standar pada *smart grid*. Sehingga, kebocoran data dapat diantisipasi dengan lebih maksimal.

Pada penelitian ini, informasi kelistrikan dari setiap pelanggan akan diambil dalam interval waktu tertentu. Informasi tersebut akan selalu melewati proses pengamanan berlapis untuk bisa terkirim ke pusat



Gambar 3.5 Gambaran dasar pertukaran data yang terjadi pada jaringan *smart grid*

pengatur data. Hal tersebut dikarenakan informasi pelanggan harus bersifat rahasia dan hanya dapat diketahui oleh pelanggan serta perusahaan yang berwenang. Selain itu, pusat pengatur juga dapat mengirimkan suatu informasi kembali ke pelanggan. Informasi tersebut dapat memiliki kegunaan yang beragam, seperti perintah khusus untuk pelanggan dan pemberitahuan tagihan pelanggan.

Pada penelitian ini, proses pertukaran data disimulasikan menggunakan aplikasi MATLAB. Semua proses pertukaran data hanya ditampilkan dalam bentuk kode program yang mengolah data dari pelanggan dan diolah kembali oleh *server*, begitu juga sebaliknya. Hal tersebut dikarenakan penelitian ini berfokus pada waktu dari proses yang dilakukan oleh sistem yang baru dibuat dalam penelitian ini. Meskipun merupakan hal yang pasti terjadi, proses yang harus dilewati oleh informasi kelistrikan selama pada jaringan *smart grid* tidak menjadi cakupan dalam penelitian ini.

3.3 Kode Program MATLAB

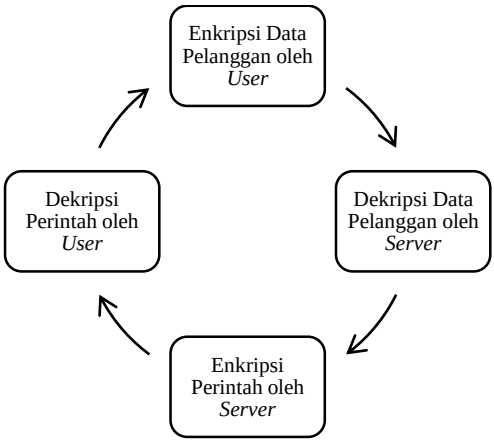
Bagian terakhir dari penyusunan program dalam penelitian ini adalah menyatukan semua potongan fungsi menjadi program yang dapat merepresentasikan jalannya data pada jaringan *smart grid*. Program yang dihasilkan dalam penelitian akan terbagi menjadi dua. Hal tersebut disesuaikan dengan dua fungsi yang bekerja independen pada jaringan.

Program pertama akan merepresentasikan jalannya data dari pengguna menuju *server* dan juga sebaliknya. Program pertama ini digunakan untuk melihat total waktu yang dibutuhkan untuk menjalankan satu siklus kegiatan saat pengambilan data ditambah permintaan data. Waktu komputasi dari program akan terpengaruh oleh variabel dalam program yang dapat diubah-ubah. Hasil yang didapatkan akan menjadi

acuan untuk perbandingan dengan frekuensi pengambilan data pada penelitian *smart grid* yang telah dilakukan sebelumnya.

Dalam program pertama ini, ada beberapa informasi atau variabel yang didefinisikan terlebih dahulu, yaitu:

a. *Raw data string* memuat beberapa informasi yang dipisahkan oleh tanda pagar. Informasi tersebut berupa enam karakter **identifikasi pelanggan** (karakter yang digunakan: A-Z dan 0-9), sepuluh digit ***unix time***, **pemakaian listrik** selama satu interval dalam satuan Wh maksimal enam karakter, dan **daya terukur** dalam satuan Watt maksimal tujuh karakter. Jumlah karakter maksimal yang terdapat pada *string* sebanyak 32 karakter. Contoh dari *string* yang akan diambil:
   `AP24V0#1543509173#123456#1234.56`
b. Pengambilan data pada jaringan *smart grid* diasumsikan akan dilakukan setiap **satu menit**.
c. Ukuran kunci RSA yang digunakan untuk rumah adalah **1536 bit** dan untuk pabrik adalah **3072 bit**.
d. Keluaran dari fungsi enkripsi berupa matriks baris dari bilangan yang sebanding dengan kunci. Sehingga, keluaran untuk rumah berukuran **1 x 192 x 8-bit *signed integer*** dan pabrik berukuran **1 x 384 x 8-bit *signed integer***.



Gambar 3.6 Siklus proses yang disimulasikan pada program RSA

e. Satu siklus didefinisikan dengan pengiriman data dari tiap pengguna menuju server dan sebaliknya. Meskipun data yang dikirimkan oleh *server* dapat berisi informasi penting, tetapi *data string* dalam penelitian ini didefinisikan hanya mengandung **identifikasi pelanggan**, **kode perintah**, dan karakter pengisi dengan jumlah karakter sebanyak 32 karakter. Contoh dari *string* yang akan dikirim:
`AP24X1#25#XXXXXXXXXXXXXXXXXXXXXX`

Rancangan program dinyatakan dapat bekerja dengan baik ketika satu siklus kegiatan dari program lebih cepat dari frekuensi pengambilan data pada jaringan *smart grid*. Selain itu, data yang didapatkan juga dapat menunjukkan gambaran dari jumlah pengguna terbanyak yang dapat dilayani oleh satu *server*.

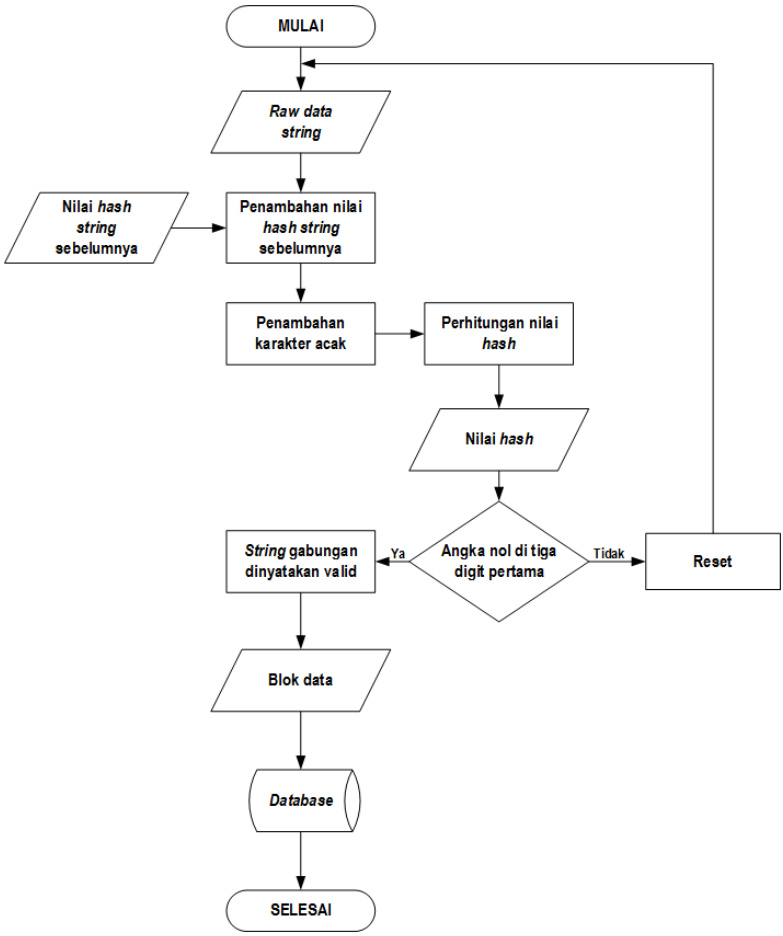
Program kedua sendiri merepresentasikan proses penyusunan blok dari *blockchain* oleh *server*. Program ini disendirikan karena program ini akan memerlukan waktu yang lebih lama jika dibandingkan dengan program sebelumnya. Hal ini dikarenakan adanya ketentuan yang dibuat untuk nilai *hash* yang diinginkan.

Sebagai contoh, untuk ketentuan angka nol di tiga digit pertama, keluaran nilai *hash* yang diperbolehkan adalah seperti berikut:
`000D829D87F43AB692C30BCC42B45FAAD22053B78161BBDBEFBF375DFA9FBE97`.

Namun, keluaran nilai *hash* yang tidak diperbolehkan adalah seperti berikut:
`25A57FD952543A1B6AE75FBE098C985C9E51BAB32CF9018EF390BE83969417EB`.

Seperti yang terlihat pada diagram alir (**Gambar 3.7**), jika program menghasilkan keluaran nilai *hash* yang tidak diperbolehkan, program harus mencari tambahan karakter lainnya secara berulang-ulang hingga nilai *hash* dari keseluruhan *string* memenuhi ketentuan. Ketentuan yang dimaksud adalah jumlah digit nol yang mengawali nilai *hash* yang dihasilkan. Ketentuan tersebut juga dapat dirubah sesuai dengan keinginan pembuat sistem, seperti jumlah digit karakter sama yang diperbanyak atau jenis karakter yang diganti selain digit nol.

Ketika program berhasil menemukan nilai *hash* yang sesuai ketentuan, *string* gabungan yang merupakan sumber dari nilai *hash* tersebut dinyatakan valid dan akan menghasilkan blok data yang baru. Blok data ini yang akan tersimpan dalam *database* tersendiri pada pusat pengatur sistem.



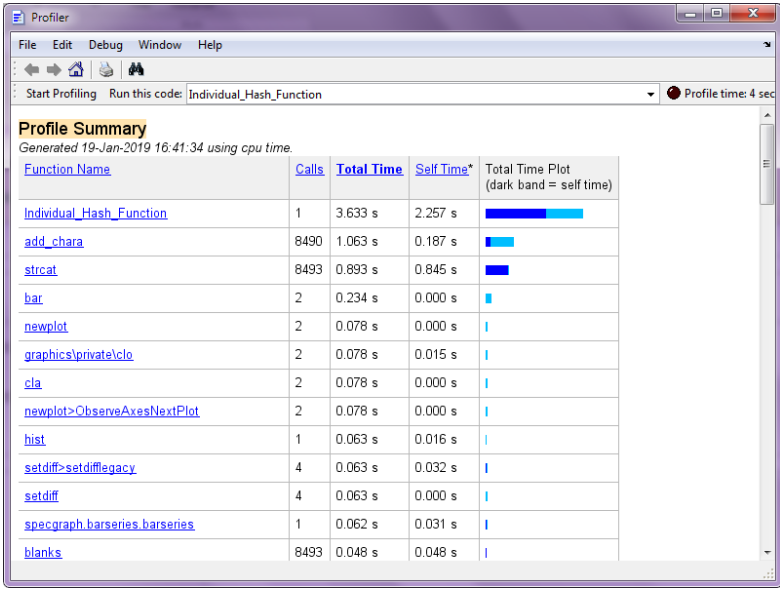
Gambar 3.7 Diagram alir penyusunan blok dari *blockchain*

Program yang dibuat juga memenuhi dasar dari *blockchain* yaitu peletakan nilai *hash* blok sebelumnya pada blok yang sedang diproses. Pada penelitian ini, program dibuat untuk mengambil 32 digit awal pada nilai *hash* sebelumnya untuk diletakkan pada *raw data string*. Ketentuan tersebut juga dapat dirubah sesuai dengan keinginan pembuat sistem, seperti jumlah karakter yang diletakkan diperkecil atau menggunakan format khusus seperti pada Bitcoin [26].

3.4 Pengujian Waktu Komputasi

Pada penelitian ini, waktu komputasi menjadi variabel utama yang diuji. Setiap kode program akan dijalankan selama beberapa kali dan waktu yang dibutuhkan setiap penyelesaian akan dicatat. Pencatatan waktu komputasi pada aplikasi MATLAB dapat dilakukan dengan dua cara, yaitu menggunakan "Run and Time" atau fungsi `tic` dan `toc`.

"Run and Time" adalah salah satu pilihan pada aplikasi MATLAB untuk menjalankan kode program sekaligus untuk mengetahui waktu berjalannya program. Pilihan ini dapat ditemukan pada jendela *Editor* dan pada tab dengan nama yang sama. Ketika kode program yang telah dibuka dijalankan menggunakan pilihan tersebut, jendela bernama *profiler* (**Gambar 3.8**) akan muncul dan menampilkan informasi lebih lanjut terkait waktu komputasi tiap baris kode program. Beberapa informasi yang dapat diperoleh pada jendela *profiler* tersebut adalah: total waktu dari kode program dan seluruh fungsi yang ada di dalamnya, jumlah pemanggilan fungsi yang telah dilakukan, serta waktu program telah selesai dijalankan.



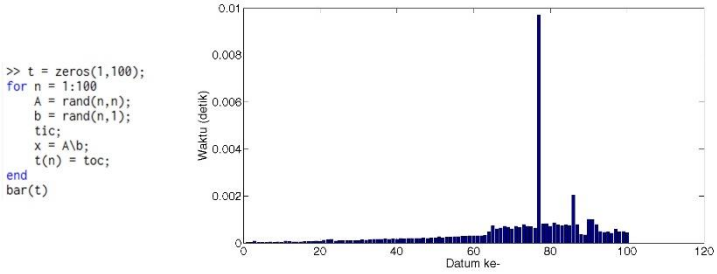
Gambar 3.8 Tampilan jendela *profiler*

Pilihan ini sejatinya memiliki nama tersendiri, yaitu *profiling*. Proses ini merupakan cara untuk mengetahui letak fungsi pada program yang menghabiskan waktu banyak. Setelah pengguna mengetahui hal tersebut, pengguna dapat mengevaluasinya untuk dapat meningkatkan performa program. Selain itu, *profiling* dapat mendeteksi galat yang terjadi di dalam sebuah program [27].

Kelemahan dari cara sebelumnya yang menggunakan "Run and Time" adalah pengguna harus mencatat secara manual hasil waktu komputasi yang diperoleh. Oleh karena itu, cara kedua yang tersedia dan yang digunakan dalam penelitian ini adalah dengan menggunakan fungsi `tic` dan `toc`.

Fungsi `tic` dan `toc` merupakan pasangan fungsi yang dapat menghasilkan keluaran berupa waktu komputasi kode program yang diapit oleh keduanya. `Tic` menjadi fungsi penanda untuk memulai perhitungan waktu, sedangkan `toc` menjadi fungsi penanda untuk mengakhiri perhitungan waktu. Fungsi `toc` dapat ditambah dengan sebuah variabel tersendiri untuk menampung keluaran berupa waktu komputasi. Keunggulan dari penggunaan pasangan fungsi ini adalah pengguna dapat memodifikasinya untuk bisa dilakukan berulang kali dalam sekali jalan. Pengulangan tersebut dapat dilakukan menggunakan fungsi lainnya, yaitu `for`.

Langkah pertama yang dilakukan adalah meletakkan pasangan fungsi `tic` dan `toc` beserta kode program yang akan diuji ke dalam fungsi `for`. Selanjutnya, nilai yang menjadi indeks `for` dapat menjadi indeks variabel untuk fungsi `toc`. Variabel tersebut telah didefinisikan sebelumnya sebagai wadah untuk menampung keluaran yang dihasilkan. Sehingga, hasil akhir yang didapatkan adalah *array* waktu komputasi dari setiap kali jalannya program.



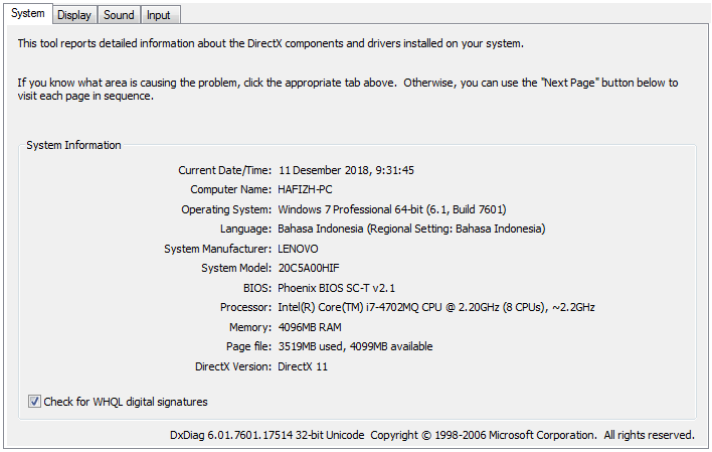
Gambar 3.9 Contoh penggunaan fungsi *tic* dan *toc* di dalam fungsi *for* pada MATLAB (kiri) beserta keluaran yang disajikan dalam bentuk grafik (kanan)

3.5 Perangkat yang Digunakan

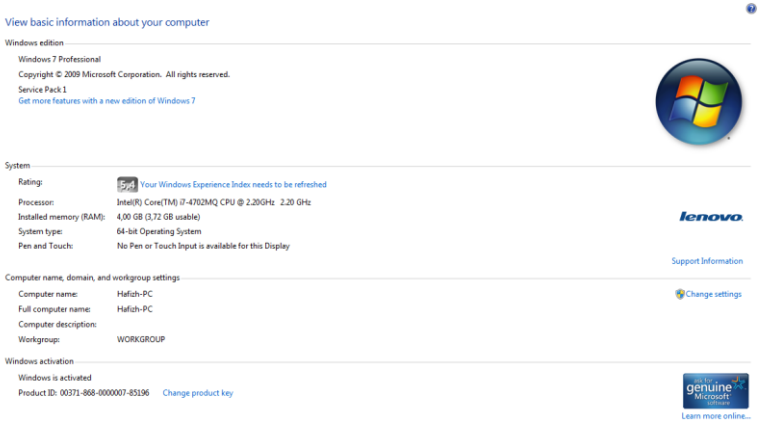
Seperti yang terlihat pada **Gambar 3.4** sebelumnya, program dari penelitian ini akan diletakkan pada perangkat milik pengguna dan perangkat *server*. Perangkat yang dimiliki pengguna tersebut dapat berupa *smart meter* yang berfungsi layaknya sebuah komputer sederhana. Di sisi lain, perangkat *server* dapat berupa komputer super yang memiliki kekuatan komputasi lebih baik dari komputer biasa.

Oleh karena itu, simulasi program dilakukan pada laptop pribadi milik penulis sebagai penengah antara dua spesifikasi komputer yang saling berlainan tersebut. Pemilihan perangkat tersebut juga memberikan kemudahan dalam melakukan simulasi karena perangkat menjalankan sistem operasi Windows 7 yang telah banyak digunakan dan aplikasi MATLAB telah kompatibel dengan sistem operasi tersebut.

Laptop milik penulis adalah Lenovo ThinkPad E440. Perangkat tersebut memiliki RAM sebesar 4 GB dan dilengkapi dengan prosesor Intel® Core™ i7 dengan seri 4702MQ yang memiliki 8 CPU dengan kecepatan masing-masing 2,2 GHz per CPU. Spesifikasi dari *Random Access Memory* (RAM) dan jenis prosesor perlu disebutkan karena MATLAB akan melakukan komputasi berulang-ulang sehingga keduanya dapat memengaruhi data yang dihasilkan.



Gambar 3.10 Cuplikan layar spesifikasi laptop milik penulis melalui DxDiag



Gambar 3.11 Cuplikan layar informasi dasar dari laptop milik penulis

Halaman ini sengaja dikosongkan

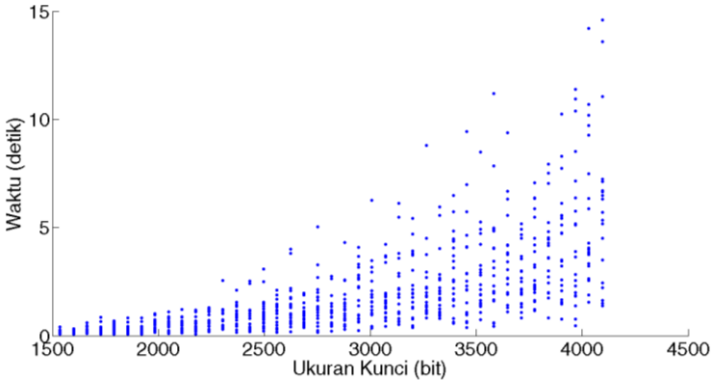
BAB 4
ANALISIS DAN PEMBAHASAN DATA

Bab ini berisikan analisis beserta pembahasan dari data yang telah didapatkan setelah simulasi program berhasil dijalankan pada perangkat. Data yang disajikan pada bab ini merupakan rangkuman dari data yang diperoleh saat penelitian dan hampir seluruhnya dalam bentuk histogram. Namun, histogram tidak bisa menampilkan tiap datum yang berhasil didapatkan selama penelitian. Oleh karena itu, data yang disajikan dalam bentuk kasar diletakkan pada bagian Lampiran sebagai kelengkapan dari penelitian. Data kasar ini juga sebagai bukti bahwa datum yang diperoleh bersifat acak dan tidak dapat diprediksi.

4.1 Ukuran Kunci RSA

Untuk mengukur waktu komputasi yang dipengaruhi oleh ukuran kunci program RSA, program dimodifikasi untuk menjalankan beberapa kali proses enkripsi dan dekripsi informasi listrik pelanggan dengan ukuran kunci yang berbeda-beda. Data yang didapatkan disajikan dalam grafik titik yang tersaji pada **Gambar 4.1**.

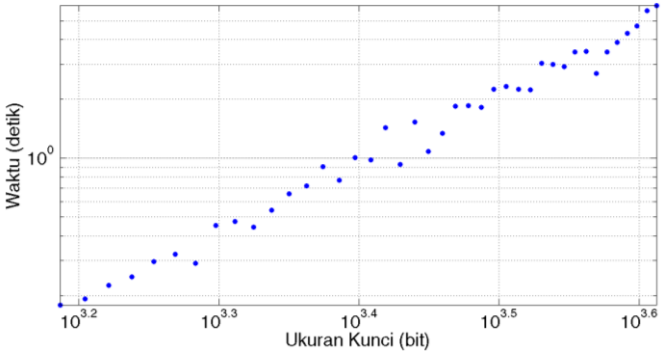
Gambar 4.1 memberikan gambaran persebaran data yang didapatkan untuk beberapa ukuran kunci RSA. Sumbu x pada grafik menunjukkan besar ukuran kunci dalam satuan bit dan sumbu y menunjukkan waktu komputasi dalam satuan detik. Setiap nilai ukuran kunci dilakukan pengukuran sebanyak 20 kali. Sedangkan rentang nilai



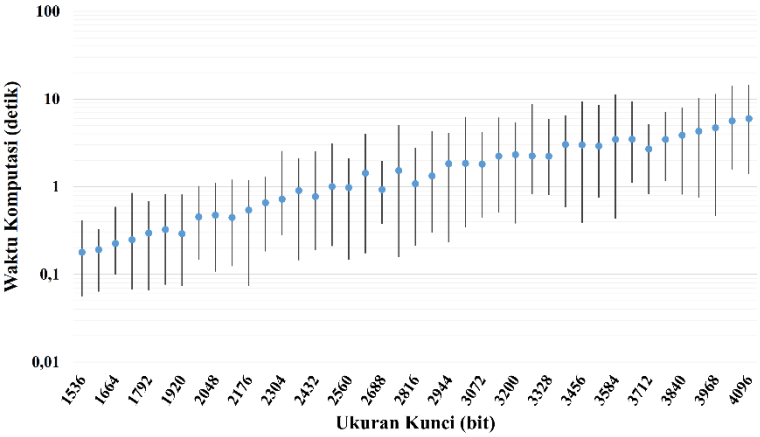
Gambar 4.1 Grafik persebaran waktu komputasi beberapa ukuran kunci RSA

ukuran kunci yang diuji adalah dari 1536 bit hingga 4092 bit dengan interval 64 bit. Oleh karena itu, titik data yang didapatkan penelitian ini berjumlah 800 titik.

Dari **Gambar 4.1**, terlihat bahwa dengan semakin besarnya ukuran kunci mengakibatkan waktu komputasi cenderung lebih bervariasi. Selain itu, pergerakan nilai rata-rata waktu komputasi menunjukkan tren naik di setiap ukuran kunci. Grafik yang menunjukkan pergerakan rata-rata waktu komputasi terdapat pada **Gambar 4.2** dan **Gambar 4.3**.



Gambar 4.2 Grafik "log-log" antara ukuran kunci dengan rata-rata waktu komputasi



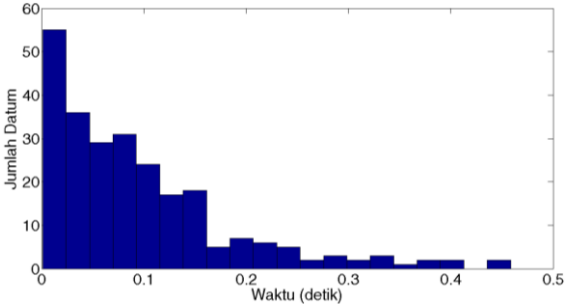
Gambar 4.3 Grafik "semi-log-y" antara ukuran kunci dengan rata-rata waktu komputasi dan dilengkapi garis rentang nilai minimum dan maksimum

Grafik "log-log" merupakan grafik yang menggunakan skala logaritmik di kedua sumbunya. Suatu fungsi monomial ($y = ax^k$) akan tampak sebagai garis lurus pada grafik "log-log". Pada **Gambar 4.2**, hubungan antara ukuran kunci dan waktu rata-rata terlihat naik secara lurus. Meskipun terjadi fluktuasi pada kenaikan yang terjadi, hal tersebut masih dapat menjadi bukti bahwa ukuran kunci RSA berpengaruh secara eksponensial terhadap waktu komputasi yang dibutuhkan. Grafik nilai minimum dan maksimum dari waktu komputasi pada **Gambar 4.3** juga terlihat bergerak naik mengikuti nilai rata-rata. Yang menjadi pembeda, grafik tersebut hanya berupa grafik "semi-log-y" atau hanya sumbu y yang memakai skala logaritmik.

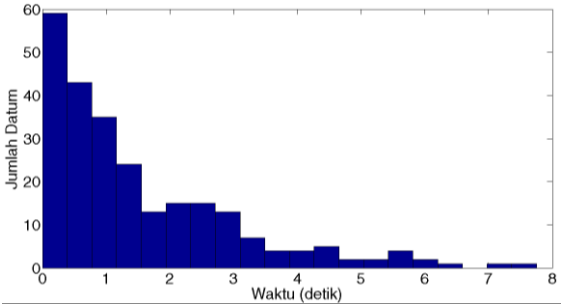
Hal lainnya yang dapat diperhatikan dari hasil yang diperoleh adalah nilai rata-rata untuk penggunaan ukuran kunci yang dipakai pada perumahan dan pabrik. Untuk ukuran kunci perumahan sebesar 1536 bit, rata-rata waktu yang dibutuhkan untuk komputasi adalah 0,1788 detik. Sedangkan untuk ukuran kunci pabrik sebesar 3072 bit, rata-rata waktu yang dibutuhkan untuk komputasi adalah 1,809 detik atau sekitar sepuluh kali lebih lama jika dibandingkan dengan perumahan.

4.2 Ketentuan Nilai *Hash*

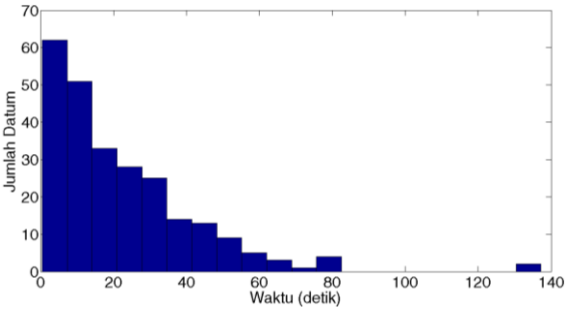
Untuk mengetahui pengaruh ketentuan nilai *hash* pada proses penyusunan *blockchain*, program *hash* pada penelitian ini diuji dengan menggunakan ketentuan yang berbeda. Pengujian dilakukan dengan menghitung waktu yang dibutuhkan untuk mencari satu nilai *hash* yang memenuhi ketentuan. Ketentuan yang dipilih pada pengujian adalah jumlah digit nol paling depan sebanyak dua, tiga, dan empat digit. Setiap ketentuan dijalankan sebanyak 250 kali. Hasil dari pengujian disajikan dalam bentuk histogram pada **Gambar 4.4 – 4.6**.



Gambar 4.4 Histogram untuk ketentuan dua digit nol



Gambar 4.5 Histogram untuk ketentuan tiga digit nol

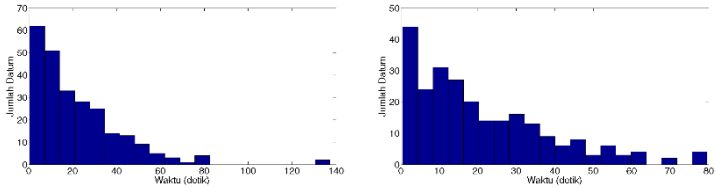


Gambar 4.6 Histogram untuk ketentuan empat digit nol

Histogram merupakan model grafik yang mengelompokkan satu set data ke dalam beberapa rentang nilai dan menampilkannya ke bentuk grafik batang. Berbeda dengan grafik batang biasa, histogram hanya berlaku untuk menampilkan satu variabel saja untuk diletakkan pada sumbu x. Sumbu y pada histogram menunjukkan banyaknya datum yang berada pada suatu rentang nilai. Histogram digunakan untuk mengetahui jenis distribusi dari nilai-nilai pada suatu set data.

Dari data yang telah didapatkan, terlihat bahwa seluruh data bersifat condong kanan (*skewed right*). Meskipun secara visual terlihat condong ke arah kiri, sifat condong kanan ini didefinisikan sebagai data yang memiliki "ekor" kanan yang lebih panjang. Distribusi data pada sifat ini terpusat di sebelah kiri grafik.

Hal lain yang dapat diperhatikan adalah adanya datum yang bersifat *outlier* di **Gambar 4.6**. Suatu datum dikatakan bersifat *outlier* jika secara visual terlihat terpisah dari kumpulan datum utama. Jika datum *outlier* terletak jauh dari kumpulan datum utama dan mengganggu



Gambar 4.7 Perbandingan histogram yang mengandung *outlier* dan yang tidak menyertakannya

penyajian, *datum* tersebut dapat dihilangkan dari penyajian data pada histogram. Perbandingannya bisa terlihat pada **Gambar 4.7**. Histogram di sebelah kanan telah dimodifikasi dengan tidak menyertakan datum yang bersifat *outlier*.

Beberapa nilai penting selama simulasi program terangkum pada **Tabel 4.1**. Waktu minimum menunjukkan waktu komputasi tercepat dari data yang didapatkan, sedangkan waktu maksimum menunjukkan waktu komputasi terlama. Simpangan baku menunjukkan tingkat variasi atau persebaran dari kumpulan datum. Nilai simpangan baku yang rendah menandakan nilai datum secara keseluruhan tidak terpaut jauh dari rata-ratanya.

Terlihat bahwa dengan semakin banyaknya digit nol yang menjadi ketentuan awalan nilai *hash*, rata-rata waktu komputasi yang dibutuhkan juga semakin lama. Selain itu, dengan bertambahnya satu digit nol, rata-rata waktu yang didapatkan juga terlihat naik dengan kelipatan tertentu.

Tabel 4.1 Rangkuman pengaruh ketentuan nilai *hash* terhadap waktu komputasi

	2 digit nol	**3 digit nol**	**4 digit nol**
Waktu Minimum	0,001114	0,000808	0,470296
Rata-rata	0,096102	1,53117	21,98469
Waktu Maksimum	0,458466	7,751967	137,1826
Simpangan Baku	0,090674	1,517616	20,37014

*angka dalam satuan detik

4.3 Kinerja Program RSA

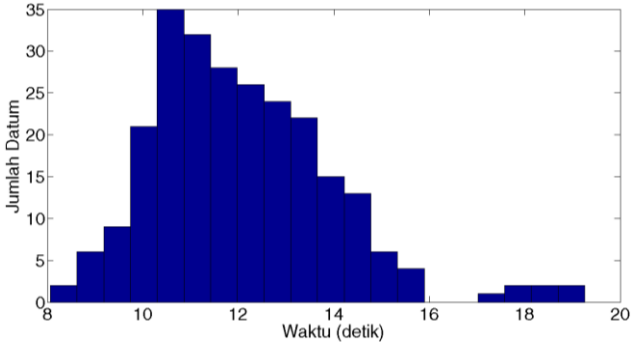
Program RSA dalam penelitian ini tersusun oleh 50 kali proses enkripsi dan dekripsi untuk pengiriman informasi listrik pelanggan dari pengguna ke *server* beserta 50 kali proses yang sama untuk pengiriman kembali informasi perintah dari *server* ke pengguna. Program telah dijalankan sebanyak 250 kali dan data yang didapatkan tersaji pada **Tabel 4.2** dan **Gambar 4.8**.

Dalam penelitian yang telah dilakukan, program berhasil melakukan proses enkripsi dan dekripsi dari seluruh data. Hal tersebut dikarenakan pesan yang dikirimkan sudah sama persis dengan pesan yang diterima. Sebagai contoh, data pelanggan yang dikirimkan adalah "`36E6WD#1541955600#77.614#329.585`" (tanpa tanda petik). Setelah melalui proses enkripsi, data tersebut berubah menjadi barisan angka seperti berikut:

[-126 35 3 -71 127 45 -118 72 -55 75 120 -25 -55 -104 -42 -126 42 48 -111 46 67 1 68 26 -3 123 -65 79 -56 -81 -31 -21 67 -29 77 127 81 -86 -19 88 55 -81 -31 94 54 54 76 90 -94 -5 -96 96 -21 -124 -102 -48 23 -101 -37 34 22 93 6 -86 -110 25 -87 -95 1 88 43 -44 11 46 -56 98 61 -46 90 47 -6 116 -18 73 90 -27 -55 39 7 126 -65 -53 -19 -80 -60 49 59 -12 3 10 -48 -30 -39 -104 -15 62 3 -108 -115 104 115 -54 69 -99 19 -33 -111 59 -38 43 -23 -63 -60 -27 -4 77 1 41 117 37 121 11 -61 -81 1 97 -1 -95 74 98 1 -43 70 -41 -128 -38 109 31 31 109 -84 73 36 73 53 35 -125 -5 14 34 53 102 113 74 -29 90 0 -101 70 62 114 -111 76 20 -12 -119 -86 86 103 -77 -31 -52 -35 87 -37 -39 113 -102 -82 -79 -71 -38]

Tabel 4.2 Rangkuman data simulasi program RSA

Waktu Minimum	Waktu Maksimum	Rata-rata	Simpangan Baku
8,0604 detik	19,2570 detik	12,0723 detik	1,9086 detik



Gambar 4.8 Histogram simulasi program RSA

Barisan tersebut memiliki format *8-bit signed integer* sehingga memiliki rentang nilai dari -128 hingga 127 jika ditampilkan dalam bentuk desimal. Data tersebut dihasilkan dari penggunaan kunci untuk perumahan yang berukuran 1536 bit. Data tersebut mengandung 192 angka, sehingga memiliki ukuran sebesar 192 *bytes* dan sebanding dengan ukuran kunci yang digunakan.

Informasi tersebut diolah kembali melalui proses dekripsi dengan menggunakan kunci yang berbeda. Namun, kedua kunci yang digunakan tetap merupakan satu pasangan. Hasil akhir dari proses dekripsi adalah "`36E6WD#1541955600#77.614#329.585`" (tanpa tanda petik) atau sama dengan data awal yang dikirimkan.

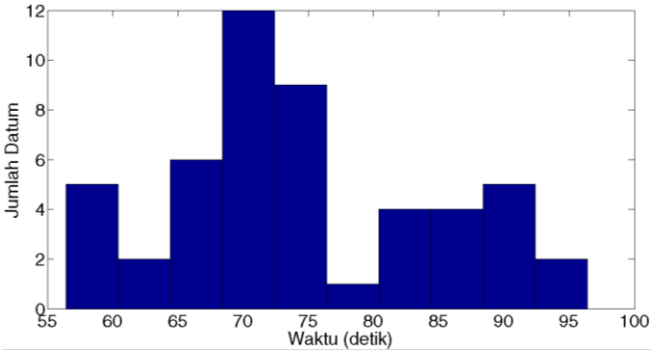
Hal yang hampir sama juga terjadi pada penggunaan kunci pabrik yang berukuran 3072 bit. Proses enkripsi dan dekripsi berhasil mengembalikan informasi yang dikirimkan. Namun, jika dibandingkan dengan proses pada perumahan, ukuran data yang dienkripsi pada pabrik menjadi lebih besar, yaitu berukuran 384 angka *8-bit signed integer* atau 384 *bytes*. Dikarenakan informasi kunci pabrik dan kunci perumahan berukuran besar, contoh keduanya tercantum pada bagian Lampiran.

Gambar 4.8 memberikan gambaran persebaran data yang didapatkan selama simulasi program RSA pada jaringan. Dari gambar tersebut, terlihat bahwa data juga bersifat condong kanan. Selain itu, terdapat beberapa datum yang bersifat ***outlier*** di bagian kanan grafik.

Setiap kali pengambilan data dilakukan, program RSA akan berjalan pada sistem. Jika pengambilan data pada jaringan *smart grid* diasumsikan akan dilakukan setiap satu menit, program dapat dinyatakan layak untuk diterapkan pada model jaringan yang telah ditentukan. Hal tersebut dikarenakan rentang waktu yang didapatkan dari pengukuran masih lebih cepat dari frekuensi pengambilan data yang ditentukan.

4.4 Kinerja Program *Blockchain*

Program *blockchain* dalam penelitian ini tersusun oleh 50 buah *string* yang disusun untuk menghasilkan *blockchain* sederhana. Nilai *hash* dari blok pertama atau *genesis block* diatur menjadi seluruhnya berupa nol sebanyak 64 digit. Nilai tersebut menjadi acuan untuk pembentukan blok selanjutnya. Pembentukan blok pada program ini menggabungkan nilai *hash* dari blok sebelumnya, informasi listrik pelanggan, dan tambahan karakter acak. Program telah dijalankan sebanyak 50 kali dan data yang didapatkan tersaji pada **Tabel 4.3** dan **Gambar 4.9**.



Gambar 4.9 Histogram simulasi program *blockchain*

Tabel 4.3 Rangkuman data simulasi program *blockchain*

Waktu Minimum	Waktu Maksimum	Rata-rata	Simpangan Baku
56,4208 detik	96,3993 detik	74,6123 detik	10,4414 detik

Dalam penelitian yang telah dilakukan, program berhasil untuk menyusun *blockchain* sederhana. Sebagai contoh, berikut ini adalah tiga blok pertama yang menjadi keluaran program:

```
00000000000000000000000000000000#UWTYK6#1541955600#437.57#
2335.16#GLLL44B3E2A0PO7FOAR8JDXRU37A4I
00044A4ACE6E68DFDBE76021CB51B921#YJDK4X#1541955600#60.782#
331.027#OQURQNWPW1LC8OFHJPOXAR5C5F55EI
000C3530C0BF3D217D24A8D509D84CF2#MES6WX#1541955600#4.9992#
57.3303#NO2D9T211KPSELGYJC29V1IBCO0D7U
```

Seperti yang telah didefinisikan sebelumnya, 32 digit pertama pada blok tersebut merupakan 32 digit pertama dari nilai *hash* blok sebelumnya. Untuk blok pertama, nilai tersebut didefinisikan berupa digit nol secara keseluruhan. Selanjutnya, nilainya kembali mengikuti aturan yang ditetapkan. Sebagai pembuktian, berikut adalah nilai *hash* SHA-256 dari blok yang pertama:

```
00044A4ACE6E68DFDBE76021CB51B9213BAE644BB23442E32FD4AEC23F1AC5EC
```

Terlihat bahwa 32 digit pertama nilai *hash* (dicetak tebal) telah sama dengan 32 digit pertama yang terdapat pada blok kedua.

Setelah potongan nilai *hash* sebelumnya, terdapat tanda pagar yang menjadi pemisah tiap informasi yang terdapat dalam setiap blok. Informasi berikutnya adalah *raw data string* yang tersusun atas kode

pelanggan, kode waktu, besar pemakaian listrik, dan besar daya listrik. Setelahnya, terdapat karakter acak yang menjadi tambahan data sehingga nilai *hash* satu blok secara keseluruhan dapat memenuhi ketentuan tiga digit pertamanya bernilai nol. Secara keseluruhan, satu blok dalam *blockchain* sederhana ini tersusun dari 96 karakter termasuk tanda pagar.

Gambar 4.9 memberikan gambaran persebaran data yang didapatkan selama simulasi program *blockchain* pada jaringan. Pengambilan data yang dilakukan hanya sebanyak 50 kali dikarenakan kepastian yang didapatkan dari hasil simulasi program *hash* untuk tiap individu sebelumnya. Program *blockchain* yang digunakan pada jaringan menggunakan ketentuan tiga digit nol pada awal nilai *hash*. Waktu rata-rata yang didapatkan untuk tiap individu adalah sekitar 1,53 detik. Jika waktu tersebut dikalikan dengan 50, hasilnya adalah sekitar 75 detik. Nilai yang dihasilkan terbukti mendekati rata-rata waktu komputasi pada **Tabel 4.3**. Waktu tersebut telah melampaui asumsi pengambilan data pada jaringan yang dilakukan setiap satu menit.

Namun, lamanya waktu menjalankan program *blockchain* tersebut tidak bisa menggambarkan keadaan yang sebenarnya jika diterapkan secara langsung pada suatu jaringan *smart grid*. Hal tersebut dikarenakan program penyusunan *blockchain* dapat dilakukan oleh perangkat yang mempunyai spesifikasi lebih baik dari komputer pada umumnya. Perangkat tersebut memang dikhusukan untuk melakukan pekerjaan dengan intensitas tinggi seperti pada *server*.

Halaman ini sengaja dikosongkan

BAB 5
PENUTUP

5.1 Kesimpulan

Kesimpulan yang dapat diambil setelah penelitian telah selesai dilakukan adalah sebagai berikut:

1. Ukuran kunci pada sistem kriptografi RSA berpengaruh secara eksponensial terhadap waktu komputasi yang dibutuhkan. Selain itu, besarnya ukuran kunci yang terlalu kecil atau terlalu besar tidak disarankan untuk digunakan secara praktis.
2. Perhitungan nilai *hash* pada pembentukan *blockchain* membutuhkan waktu yang semakin lama seiring dengan semakin sulitnya ketentuan yang perlu dipenuhi. Pengaruh penambahan satu digit yang berulang pada ketentuan mengakibatkan perubahan yang signifikan terhadap waktu pencarian nilai *hash*.
3. Penerapan program RSA dan *blockchain* dapat diterapkan pada model jaringan *smart grid* yang memiliki topologi bintang atau pohon. Di bidang kelistrikan, distribusi listrik pada *smart grid* menggunakan pola radial.
4. Peletakan program RSA pada *application layer* memungkinkan program dapat diletakkan pada berbagai jenis jaringan seperti TCP/IP. Selain itu, pembentukan *blockchain* yang hanya dikhusukan pada suatu perangkat memungkinkan adanya pembagian tanggung jawab dalam penggunaan data.
5. Kriptografi menggunakan RSA dapat memberikan keamanan pada jaringan *smart grid* dengan merahasaikan transmisi data antara pelanggan dan *server*. Di sisi lain, pembentukan *blockchain* dapat memberikan keamanan tambahan saat penyimpanan data dalam *database*.
6. Simulasi program yang telah dijalankan perlu dikembangkan dengan lebih lanjut pada jaringan *smart grid* lainnya karena adanya perbedaan perangkat dan aplikasi yang akan digunakan. Aplikasi MATLAB yang dipakai pada penelitian ini tidak bisa diterapkan secara langsung pada beberapa perangkat. Selain itu, jenis perangkat yang digunakan pada jaringan *smart grid* akan memiliki spesifikasi yang berbeda dengan perangkat yang digunakan selama penelitian.

5.2 Saran

Saran untuk pengembangan implementasi program kriptografi dan *blockchain* pada jaringan *smart grid* adalah sebagai berikut:

1. Pengujian sistem kriptografi lainnya seperti *Advanced Encryption Standard* (AES) dapat dilakukan untuk mengetahui waktu komputasi yang dibutuhkan.
2. Jenis perangkat yang digunakan perlu lebih menyesuaikan dengan kondisi sebenarnya pada suatu jaringan *smart grid*.
3. Program perlu disusun ulang dengan tidak menggunakan MATLAB agar bisa diterapkan di lebih banyak jenis perangkat.
4. Simulasi secara *real-time* perlu dilakukan antara dua perangkat dengan menggunakan protokol tertentu seperti TCP/IP.
5. Perlunya simulasi dari model jaringan yang berbeda untuk beberapa perangkat yang terhubung dalam satu jaringan.
6. Pembentukan *blockchain* yang lebih kompleks perlu dibuat untuk bisa memuat lebih banyak informasi.

DAFTAR PUSTAKA

[1] J. Gao, K. O. Asamoah, E. B. Sifah, A. Smahi, Q. Xia, H. Xia, X. Zhang dan G. Dong, "GridMonitoring: Secured Sovereign Blockchain Based Monitoring on Smart Grid," *IEEE Access*, vol. 6, pp. 9917-9925, 2018.

[2] National Institute of Standards and Technology, *NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 3.0,* National Institute of Standards and Technology, 2014.

[3] S. V. Nandury dan B. A. Begum, "Big Data for Smart Grid Operation in Smart Cities," dalam *IEEE International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, 2017.

[4] Laboratory for Advanced System Software, "Smart - UMass Trace Repository," 17 November 2018. [Online]. Available: http://traces.cs.umass.edu/index.php/Smart/Smart. [Diakses 30 November 2018].

[5] Y. Wang, Q. Chen, T. Hong dan C. Kang, "Review of Smart Meter Data Analytics: Applications, Methodologies, and Challenges," *IEEE Transactions on Smart Grid,* 2018.

[6] The Research Perspective Ltd., *Smart Meter Electricity Trial Data Manifest,* Commission for Regulation of Utilities, 2012.

[7] B. Römer, Y. Julliard, R. Fauzianto, M.-J. Poddey dan I. Rendroyoko, "Pioneering smart grids for Indonesia – the case of a smart grid roadmap development," dalam *24th International Conference & Exhibition on Electricity Distribution (CIRED)*, Glasgow, 2017.

[8] PT PLN (Persero), *Meter Statik Energi Aktif Fase Tunggal Prabayar dengan Sistem Standard Transfer Specification (STS),* Jakarta: PT PLN (Persero), 2010.

[9] T. Flick dan J. Morehouse, Securing the Smart Grid: Next Generation Power Grid Security, Rockland: Syngress, 2011.

[10] Physicians for Safe Technology, ""Smart" Meters," Physicians for Safe Technology, 4 Desember 2017. [Online]. Available: https://mdsafetech.org/smart-meters/. [Diakses 1 November 2018].

[11] G. N. Sorebo dan M. C. Echols, Smart Grid Security: An End-to-End View of Security in the New Electrical Grid, Boca Raton, Florida: CRC Press, 2012.

[12] The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Local Area Network/Wide Area Network (LAN/WAN) Node Communication Protocol to Complement the Utility Industry End Device Data Tables*, New York: The Institute of Electrical and Electronics Engineers, Inc., 2012.

[13] C. Brunschwiler, "Advanced Metering Infrastructure Architecture and Components," Compass Security, 28 Februari 2013. [Online]. Available: https://blog.compass-security.com/2013/02/advanced-metering-infrastructure-architecture-and-components/. [Diakses 15 Januari 2019].

[14] S. P. Sicca, "Potensi Kerugian PLN Akibat Pencurian Listrik Sebesar Rp10 Triliun," Tirto.ID, 24 April 2018. [Online]. Available: https://tirto.id/potensi-kerugian-pln-akibat-pencurian-listrik-sebesar-rp10-triliun-cJlX. [Diakses 15 Januari 2019].

[15] D. Ariyus, Pengantar Ilmu Kriptografi: Teori, Analisis, dan Implementasi, Yogyakarta: Penerbit ANDI, 2008.

[16] W. Stallings, Cryptography and Network Security: Principles and Practice, 7th ed., Boston: Prentice Hall, 2011.

[17] R. L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, pp. 120-126, Januari 1978.

[18] E. Barker, *Recommendation for Key Management, Part 1: General, Revision 4*, Gaithersburg: National Institute of Standards and Technology, 2016.

[19] K. Christidis dan M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292-2303, 2016.

[20] M. Gupta, Blockchain For Dummies, IBM Limited Edition, Hoboken: John Wiley & Sons, Inc., 2017.

[21] D. Khovratovich, C. Rechberger dan A. Savelieva, "Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family," dalam *International Conference on Fast Software Encryption*, Washington, D.C., 2012.

[22] W. Stallings, Data and Computer Communications, 10th ed., New Jersey: Prentice Hall, 2015.

[23] MathWorks Inc., "MathWorks - Company - About Us - Products - MATLAB & Simulink," MathWorks Inc., [Online]. Available: https://www.mathworks.com/company/aboutus/products.html. [Diakses 1 Mei 2018].

[24] A. Janke, "encryption - RSA code in matlab," Stack Overflow, 24 Februari 2012. [Online]. Available: https://stackoverflow.com/a/9436658. [Diakses 12 Oktober 2018].

[25] G. Walker, "Learn Me a Bitcoin - Target," 18 Januari 2016. [Online]. Available: http://learnmeabitcoin.com/glossary/target. [Diakses 27 November 2018].

[26] G. Walker, "Learn Me a Bitcoin - Block Header," 23 Maret 2016. [Online]. Available: http://learnmeabitcoin.com/glossary/block-header. [Diakses 16 Januari 2019].

[27] MathWorks, Inc., "Profile to Improve Performance," MathWorks, Inc., 13 September 2018. [Online]. Available: https://www.mathworks.com/help/matlab/matlab_prog/profiling-for-improving-performance.html. [Diakses 18 Januari 2019].

Halaman ini sengaja dikosongkan

LAMPIRAN A
LEMBAR PENGESAHAN PROPOSAL

Departemen Teknik Elektro
Fakultas Teknologi Elektro – ITS

EE184801 TUGAS AKHIR – 6 SKS

05 SEP 2018

Nama Mahasiswa	:	Hafizh Fianto Putra
NRP	:	07111440000145
Bidang Studi	:	Telekomunikasi Multimedia
Tugas Diberikan	:	Semester Ganjil Tahun Akademik 2018 – 2019
Dosen Pembimbing	:	1. Dr. Ir. Wirawan, DEA. 2. Prof. Ir. H. Ontoseno Penangsang, M.Sc., Ph.D.
Judul Tugas Akhir	:	**Penerapan *Blockchain* dan Kriptografi untuk Keamanan Data pada Jaringan *Smart Grid*** ***(The Implementation of Blockchain and Cryptography for Data Security in Smart Grid Network)***

Uraian Tugas Akhir :

Smart grid memperkenalkan komunikasi dua arah antara pelanggan dengan perusahaan penyedia listrik. Namun, masalah yang dapat terjadi pada jaringan *smart grid* adalah data pelanggan yang jatuh ke pihak yang tidak bertanggungjawab karena saluran transmisi yang tidak aman. Penggunaan *blockchain* dapat diterapkan pada jaringan *smart grid* untuk menyelesaikan masalah tersebut. Penerapan kriptografi juga diperlukan untuk memberikan kerahasiaan pada pertukaran data dalam jaringan. Oleh karena itu, suatu simulasi dari jaringan *smart grid* yang aman diperlukan untuk mengetahui variabel yang efektif untuk dapat diterapkan secara nyata. Simulasi dilakukan dengan menggunakan aplikasi MATLAB untuk memudahkan pemrograman. Setelah simulasi berhasil dijalankan, evaluasi dilakukan untuk mendapatkan hasil yang terbaik.

Kata kunci : *Smart grid*, *Blockchain*, Kriptografi, MATLAB

Dosen Pembimbing I,

Dr. Ir. Wirawan, DEA.
NIP. 196311091989031011

Dosen Pembimbing II,

Prof. Ir. H. Ontoseno Penangsang, M.Sc., Ph.D.
NIP. 194907151974121001

Mengetahui,
Ketua Program Studi S1

Dedet C. Riawan, ST. M.Eng. Ph. D.
NIP. 197311192000031001

Mengetahui,
Kepala Laboratorium Komunikasi Multimedia

Dr. Ir. Endroyono, DEA.
NIP. 196504041991021001

Halaman ini sengaja dikosongkan

LAMPIRAN B
SKRIP MATLAB

Kode Program RSA Individual

```
clc;
clear all;

import javax.crypto.Cipher; %Objek Java untuk Program Ini
%Pengaturan Rentang Ukuran Kunci yang Disimulasikan
bit_start = 1536;   %Bit Mulai
bit_interval = 64;  %Jarak Antar Ukuran Bit
run_times = 20;     %Jumlah Simulasi tiap Ukuran Bit
bit_step = 40;      %Jumlah Lompatan atau Sampel Bit
bit_end = bit_start+(bit_interval*bit_step);    %Bit Akhir
%Perhitungan Ukuran Array
array_size = (bit_step+1)*run_times;
%Deklarasi Array Ukuran Kunci
key_size = zeros(1,array_size);
%Deklarasi Array Waktu Komputasi
run_time = zeros(1,array_size);

%% Pengulangan Simulasi
for bit_ID = bit_start:bit_interval:bit_end
for run_ID = 1:run_times
tic;
idx = run_times*((bit_ID-bit_start)/bit_interval)+run_ID;
key_size(idx) = bit_ID;
%% Enkripsi dari Pengguna ke Server
plaintext1 = 'AP24V0#1541955600#6.5812#479.502';
plaintextUnicodeVals1 = uint16(plaintext1);

cipher1 = Cipher.getInstance('RSA');
keygen1 =
java.security.KeyPairGenerator.getInstance('RSA');
keygen1.initialize(bit_ID);
keyPair1 = keygen1.genKeyPair();
cipher1.init(Cipher.ENCRYPT_MODE, keyPair1.getPrivate());
plaintextBytes1 = typecast(plaintextUnicodeVals1, 'int8');
ciphertext1 = cipher1.doFinal(plaintextBytes1)';

%% Dekripsi oleh Server
cipher1.init(Cipher.DECRYPT_MODE, keyPair1.getPublic());
```

```
decryptedBytes1 = cipher1.doFinal(ciphertext1);
decryptedText1 = char(typecast(decryptedBytes1,
'uint16'))';

%% Enkripsi dari Server ke Pengguna
plaintext6 = 'AP24V0#25#XXXXXXXXXXXXXXXXXXXXX';
plaintextUnicodeVals6 = uint16(plaintext6);

cipher1.init(Cipher.ENCRYPT_MODE, keyPair1.getPublic());
plaintextBytes6 = typecast(plaintextUnicodeVals6, 'int8');
ciphertext6 = cipher1.doFinal(plaintextBytes6)';

%% Dekripsi oleh Pengguna
cipher1.init(Cipher.DECRYPT_MODE, keyPair1.getPrivate());
decryptedBytes6 = cipher1.doFinal(ciphertext6);
decryptedText6 = char(typecast(decryptedBytes6,
'uint16'))';
    run_time(idx) = toc;
end
end
%% Tampilan Grafik Scatter
scatter(key_size,run_time)
```

Kode Program *Hash* Individual

```
clc;
clear all;

epoch_time = 1541955600;    %Asumsi Waktu Koleksi Data
%Deklarasi Nilai Hash Blok Awal
prev_hash =
'0000000000000000000000000000000000000000000000000000000000000000
0000000';
%Objek Java untuk Program Ini
sha256hasher = System.Security.Cryptography.SHA256Managed;
run_times = 250;     %Jumlah Pengulangan Simulasi
%Deklarasi Array Waktu Komputasi
run_time = zeros(1,run_times);

%% Pengulangan Simulasi
for run_id = 1:run_times
tic;
```

```
%% Pembuatan Data String
rawdata1 = gen_string(gen_ID(),epoch_time);
string1 = add_chara(prev_hash,rawdata1);
%% Perhitungan Nilai Hash
sha256_1 =
uint8(sha256hasher.ComputeHash(uint8(string1)));
while sha256_1(1)+sha256_1(2)~=0
    string1 = add_chara(prev_hash,rawdata1);
    sha256_1 =
uint8(sha256hasher.ComputeHash(uint8(string1)));
end
hash1 = reshape(dec2hex(sha256_1)',64,1)';
run_time(run_id) = toc;
end
%% Tampilan Grafik dan Histogram
his_kel = 20;   %Jumlah Kelompok dalam Histogram
figure(1)
bar(run_time)
figure(2)
hist(run_time,his_kel)
```

Kode Program RSA Pada Jaringan *Smart Grid*

```
clc;
clear all;

import javax.crypto.Cipher;  %Objek Java untuk Program Ini
run_times = 150;             %Jumlah Pengulangan Simulasi
%Deklarasi Array Waktu Komputasi
run_time = zeros(1,run_times);
epoch_time = 1541955600;     %Asumsi Waktu Koleksi Data

%% Deklarasi Variabel Rumah
n_user1 = 48;
keysz1 = 1536;
IDPel1 = cell(1,n_user1);
cipher1 = cell(1,n_user1);
keygen1 = cell(1,n_user1);
keyPair1 = cell(1,n_user1);
plaintext1 = cell(1,n_user1);
ciphertext1 = cell(1,n_user1);
decryptedtext1 = cell(1,n_user1);
```

```
plaintext3 = cell(1,n_user1);
ciphertext3 = cell(1,n_user1);
decryptedtext3 = cell(1,n_user1);

%% Deklarasi Variabel Pabrik
n_user2 = 2;
keysz2 = 3072;
IDPel2 = cell(1,n_user2);
cipher2 = cell(1,n_user2);
keygen2 = cell(1,n_user2);
keyPair2 = cell(1,n_user2);
plaintext2 = cell(1,n_user2);
ciphertext2 = cell(1,n_user2);
decryptedtext2 = cell(1,n_user2);
plaintext4 = cell(1,n_user2);
ciphertext4 = cell(1,n_user2);
decryptedtext4 = cell(1,n_user2);

%% Pengulangan Simulasi
for run_ID = 1:run_times
tic;
%% Pembuatan Kunci Rumah
for idx = 1:n_user1
    IDPel1{idx} = gen_ID();
    cipher1{idx} = Cipher.getInstance('RSA');
    keygen1{idx} =
java.security.KeyPairGenerator.getInstance('RSA');
    keygen1{idx}.initialize(keysz1);
    keyPair1{idx} = keygen1{idx}.genKeyPair();
end

%% Pembuatan Kunci Pabrik
for idx = 1:n_user2
    IDPel2{idx} = gen_ID();
    cipher2{idx} = Cipher.getInstance('RSA');
    keygen2{idx} =
java.security.KeyPairGenerator.getInstance('RSA');
    keygen2{idx}.initialize(keysz2);
    keyPair2{idx} = keygen2{idx}.genKeyPair();
end

%% Enkripsi dari Pengguna ke Server
for idx = 1:n_user1
```

```
    plaintext1{idx} = gen_string(IDPel1{idx},epoch_time);
    plaintextUnicodeVals = uint16(plaintext1{idx});
    cipher1{idx}.init(Cipher.ENCRYPT_MODE,
keyPair1{idx}.getPrivate());
    plaintextBytes = typecast(plaintextUnicodeVals,
'int8');
    ciphertext1{idx} =
cipher1{idx}.doFinal(plaintextBytes)';
end
for idx = 1:n_user2
    plaintext2{idx} = gen_string(IDPel2{idx},epoch_time);
    plaintextUnicodeVals = uint16(plaintext2{idx});
    cipher2{idx}.init(Cipher.ENCRYPT_MODE,
keyPair2{idx}.getPrivate());
    plaintextBytes = typecast(plaintextUnicodeVals,
'int8');
    ciphertext2{idx} =
cipher2{idx}.doFinal(plaintextBytes)';
end

%% Dekripsi oleh Server
for idx = 1:n_user1
    cipher1{idx}.init(Cipher.DECRYPT_MODE,
keyPair1{idx}.getPublic());
    decryptedBytes =
cipher1{idx}.doFinal(ciphertext1{idx});
    decryptedtext1{idx} = char(typecast(decryptedBytes,
'uint16'))';
end
for idx = 1:n_user2
    cipher2{idx}.init(Cipher.DECRYPT_MODE,
keyPair2{idx}.getPublic());
    decryptedBytes =
cipher2{idx}.doFinal(ciphertext2{idx});
    decryptedtext2{idx} = char(typecast(decryptedBytes,
'uint16'))';
end

%% Enkripsi dari Server ke Pengguna
for idx = 1:n_user1
    plaintext3{idx} = gen_command(IDPel1{idx});
    plaintextUnicodeVals = uint16(plaintext3{idx});
```

```
    cipher1{idx}.init(Cipher.ENCRYPT_MODE,
keyPair1{idx}.getPublic());
    plaintextBytes = typecast(plaintextUnicodeVals,
'int8');
    ciphertext3{idx} =
cipher1{idx}.doFinal(plaintextBytes)';
end
for idx = 1:n_user2
    plaintext4{idx} = gen_command(IDPel2{idx});
    plaintextUnicodeVals = uint16(plaintext4{idx});
    cipher2{idx}.init(Cipher.ENCRYPT_MODE,
keyPair2{idx}.getPublic());
    plaintextBytes = typecast(plaintextUnicodeVals,
'int8');
    ciphertext4{idx} =
cipher2{idx}.doFinal(plaintextBytes)';
end

%% Dekripsi oleh Pengguna
for idx = 1:n_user1
    cipher1{idx}.init(Cipher.DECRYPT_MODE,
keyPair1{idx}.getPrivate());
    decryptedBytes =
cipher1{idx}.doFinal(ciphertext3{idx});
    decryptedtext3{idx} = char(typecast(decryptedBytes,
'uint16'))';
end
for idx = 1:n_user2
    cipher2{idx}.init(Cipher.DECRYPT_MODE,
keyPair2{idx}.getPrivate());
    decryptedBytes =
cipher2{idx}.doFinal(ciphertext4{idx});
    decryptedtext4{idx} = char(typecast(decryptedBytes,
'uint16'))';
end
run_time(run_ID) = toc;
end
%% Tampilan Grafik dan Histogram
his_kel = 20;   %Jumlah Kelompok dalam Histogram
figure(1)
bar(run_time)
figure(2)
hist(run_time,his_kel)
```

Kode Program *Blockchain* pada Jaringan *Smart Grid*

```
clc;
clear all;

n_user = 25;                %Jumlah Asumsi Pengguna Jaringan
run_times = 50;             %Jumlah Pengulangan Simulasi
%Deklarasi Array Waktu Komputasi
run_time = zeros(1,run_times);
epoch_time = 1541955600;  %Asumsi Waktu Koleksi Data
%Objek Java untuk Program Ini
sha256hasher = System.Security.Cryptography.SHA256Managed;

%% Deklarasi Data Array
string = cell(1,n_user);
sha256 = cell(1,n_user);
rawdata = cell(1,n_user);

%% Pengulangan Simulasi
for run_ID=1:run_times
tic;
%Deklarasi Nilai Hash Blok Awal
prev_hash =
'0000000000000000000000000000000000000000000000000000000000000000';
%% Pembuatan Raw Data String
for idx = 1:n_user
    rawdata{idx} = gen_string(gen_ID(),epoch_time);
end

%% Perhitungan Nilai Hash
for idx = 1:n_user
    string{idx} = add_chara(prev_hash,rawdata{idx});
    sha256{idx} =
uint8(sha256hasher.ComputeHash(uint8(string{idx})));
    while sha256{idx}(1)~=0 || sha256{idx}(2)>=16
        string{idx} = add_chara(prev_hash,rawdata{idx});
        sha256{idx} =
uint8(sha256hasher.ComputeHash(uint8(string{idx})));
    end
    prev_hash = reshape(dec2hex(sha256{idx})',64,1)';
end
run_time(run_ID) = toc;
end
```

```
%% Tampilan Grafik dan Histogram
his_kel = 10;   %Jumlah Kelompok dalam Histogram
figure(1)
bar(run_time)
figure(2)
hist(run_time,his_kel)
```

LAMPIRAN C
FUNGSI TAMBAHAN

Fungsi Penambahan Karakter Acak Pada Program *Hash*

```
function string = add_chara(prev,data)
symbols = ['A':'Z' '0':'9'];
stLength = 62-length(data);
nums = randi(numel(symbols),[1 stLength]);
st = symbols (nums);
prev_cut = prev(1:32);
string = strcat(prev_cut,'#',data,'#',st);
end
```

Fungsi Pembuatan Komando dari *Server*

```
function string_el = gen_command(IDPel)
command_number = randi([10 49],[1 1]);
string_el =
strcat(IDPel,'#',num2str(command_number),'#XXXXXXXXXXXXXXX
XXXXXXX');
end
```

Fungsi Pembuatan Nomor Pelanggan Secara Acak

```
function IDPel = gen_ID(~)
symbols = ['A':'Z' '0':'9'];
NPLength = 6;
NP_nums = randi(numel(symbols),[1 NPLength]);
IDPel = symbols (NP_nums);
end
```

Fungsi Pembuatan *Raw Data String* Secara Acak

```
function string_el = gen_string(IDPel,time)
number_usage1 = randi([10000 99999],[1 1]);
number_usage2 = randi([100000 999999],[1 1]);
multi = [0.0001 0.001 0.01];
index = randi([1 3],[1 1]);
usage1 = number_usage1*multi(index);
usage2 = number_usage2*multi(index);
string_el =
strcat(IDPel,'#',num2str(time),'#',num2str(usage1),'#',num
2str(usage2));
end
```

Halaman ini sengaja dikosongkan

LAMPIRAN D
CONTOH KELUARAN

Kunci RSA Ukuran 1536 bit
(beberapa digit disensor untuk alasan keamanan)

Sun RSA private CRT key, 1536 bits

modulus: ($n = p \cdot q$)

13896498117712237940051809261715814312572600008774600766700969972158848522593897974508211154970797813251404856786935601955086282086095649156978526020374710627233602285820241589330912577322841109391074723345361285436350532277487899934533098504372870318515036929478284700557508296002974440425007904602711236529771308809032661400175648828052148004742678343979234681630570557962046837691673998315933874418116735361272242733759480843925390419050460983023633528781657136553185402584397103150979172331894399221644404731029063250678575702030192458939535971880509916314417747028683557413287405295261055614230308423943931968025253484088964123780141834839993744388519049258040947925577423307119803450241263393564551908436832275838334147471011687295119368722837133242713412101282255201905307861708269065293587087954767588442062217318039980763531585633436931374271749174585558968257618923509485235146652965764650045243657283125741572332937 prime p: (p)

11915544514684962586101227459867683124081643342461751818069663590559050994415366695718224768752210628661286667831476638423330686673302289035357320852551442318826087184184146264908525124762916725295300992026715433827106956938215496697 prime q: (q)

11662495239379037453945411304072238833552531630745884630457761651088954187207908043573790430227747112905246590769425796107811026401782925061967401832974484592950633074823835209091575336138598432704206456583011685014557163064447487737 prime exponent p: ($d_P \equiv d \pmod{p-1}$)

847380538010470499489479085435789107472226606744457126485455255756573675476016594576100983867485970932475682023317563646483930167359811686636401808100561624965328662272106603855214047774272944772934515365959879494606060610066694621
prime exponent q: ($d_Q \equiv d \pmod{q-1}$)
401621763366564683000585909824383719478229223758035720623161302323003199797206580459003122236004126815628591951244082089866363589249101856313768145012496931928269960154675852851669707762227753486992269331617558130383377422402843485
crt coefficient: ($q_{inv} \equiv q^{-1} \pmod{p}$)
111990403070446708018520711191258937527976676110182500129708645245037809435837424309452715694256122389544293076302989510617732287015700734056939944643879327561600997138572235366702099252110644789442822863928135965874395328621846128 3

Kunci RSA Ukuran 3072 bit
(beberapa digit disensor untuk alasan keamanan)
Sun RSA private CRT key, 3072 bits
modulus: ($n = p \cdot q$)
339506054176229321937302971146996915568444947387160636358204599204594148425469200495642159708844479800681327563659899733007878814759529032544643227247932661082637951234577290451439130299778716123733304350320463993107770673122062560400449201846470378080348824246725160316802709416587207891071196430751507123145329926753123449750862995501037304529054770482289776205114442671917809264766443897493759612602851828145577856401812868212313822096396340294569077322441704003204827706331728462656059407206212304846859240695401865156215314286907331315838978434393456643702905499341712981751182866359335703180067298004364728699467755253027257123027096840541347139294500586172820666078223660158365855897196618050569101394823210472327188704010920069662687140860300619724097619597541391729990159854585651269840399625155420426159316901521191990209313854360717818861400124368645114388700591383226186250485607264870137180938258144473641078970765537
private exponent: ($d \equiv e^{-1} \pmod{\phi(n)}$)

24302680195262899095235812266992792315886381458983271943287688424073615339554811179718419577370067413658487754999430666912734209434081183917125971161149068357370331577990638001355515082706744143785495990030950344551421098903177321049279450396300509402144444194709274830923242606253499067387617739835276948392689416442268154474819164300990052911694393464399600267194185551530842148105343269079012579871833862980269694062495727159687547238355164489103730583399866890847635745104912964672003938838453694143294000351162664571717155967769545996449906578824800711462984793835387559689778333565454967587291304581352953860040166307024257573506341976917858976387268734533037856012744409836348121662399150910510514238983991374418556685957458995408097021099644518372336379995215355311020950577615661858036930538959720413093816625551644432500978866287352916769621432676119766764766956052844335413057523901915000786983969339327864044457257
prime p: (p)
20109069919712700515850385759927896703896417275622184284623267837319292051507865052889503330537890806068844130574709643902026680041055781355634089791852011987522639637291493923182426162840769942484077284344968148552804557640587369320220890471385857230247414869624859976291660200461353693541006367608755708214257560314376301345932605158097352613581509198948298613184644776255356441961461290053171216169639400715087238928875270606032891959395665314189309094202131399
prime q: (q)
16883230081338334977078379190095958224482443205390176112060829969201651844719377291982171053553536028630134059545670619515622145006940414739939559108889643615137263568331519008707949732635529760797507682165289146158308640689212131606218249193708479002784711386318933567662088313158876176650446610580772858675961336419085081242132548191708531888725227667197382224882745977575910370104284189623474013420961659532051395950048935141251956734753752324918195203259904931288371052777540293208480473127565275645675976907853420094063495767581295986639220144006218389544069847604706041932775741645480062016709955203503053649196837426321186093706666982783606438379982475493112428644685581555015595420332103

06659592291838314118873110298103093755361173101785522718173446506480925802555573294481033083176581802338121937057185099982397746382808192244104271120362512490355470329351341679701234532225374578446854601954719610109548651229768367913226375737671646326448928863635000993490589366794213951632933995490006330056405661889140548139693470110135819954877310812214226131761205324801666356164305782706825593699261362598196496458802692821125895344071657641107827613661694950958022180198160517595215580736687569767029221629935849531106923781943409333123611590718881819298531377308233793665861722234755357450810903989274604003172681189471300988762154148537055744510993062876908458431521805931097115263504165crt coefficient: ($q_{inv} \equiv q^{-1} \pmod{p}$)1076390028593820109270978510238391822030280788670914718281934956376509673470966927940426894858852353427393801572102264026023119289912817191279576790979658159893261218718545668874948537831070491265837522610253493385239060373485362485353264571487302554526704515404830137109133115813495227112517010055933304138629156202363451191516738975543096468605802197142019004034501884952203776926879861225448568400121131195732059272797643809847225932651654976902688487501671894

Proses Enkripsi dan Dekripsi (1536 bit)

Plain-text : 36E6WD#1541955600#77.614#329.585

Cipher-text : [-126 35 3 -71 127 45 -118 72 -55 75 120 -25 -55 -104 -42 -126 42 48 -111 46 67 1 68 26 -3 123 -65 79 -56 -81 -31 -21 67 -29 77 127 81 -86 -19 88 55 -81 -31 94 54 54 76 90 -94 -5 -96 96 -21 -124 -102 -48 23 -101 -37 34 22 93 6 -86 -110 25 -87 -95 1 88 43 -44 11 46 -56 98 61 -46 90 47 -6 116 -18 73 90 -27 -55 39 7 126 -65 -53 -19 -80 -60 49 59 -12 3 10 -48 -30 -39 -104 -15 62 3 -108 -115 104 115 -54 69 -99 19 -33 -111 59 -38 43 -23 -63 -60 -27 -4 77 1 41 117 37 121 11 -61 -81 1 97 -1 -95 74 98 1 -43 70 -41 -128 -38 109 31 31 109 -84 73 36 73 53 35 -125 -5 14 34 53 102 113 74 -29 90 0 -101 70 62 114 -111 76 20 -12 -119 -86 86 103 -77 -31 -52 -35 87 -37 -39 113 -102 -82 -79 -71 -38] (192 bilangan *signed 8-bit*)

Decrypted-text : 36E6WD#1541955600#77.614#329.585

Proses Enkripsi dan Dekripsi (3072 bit)

Plain-text : Z1JYXF#1541955600#925.47#3572.55

Cipher-text : [77 3 63 -28 -2 -127 -92 59 117 -44 -49 24 121 -110 116 -48 104 3 28 -23 -72 68 85 -94 -71 124 -37 63 15 -121 97 116 38 89 1 79 62 -83 8 107 -41 12 96 109 -121 50 112 21 35 -12 8 114 -10 89 -109 -84 -98 -11 -2 -97 81 -66 88 28 59 72 62 108 98 -21 -96 7 -15 42 -99 -77 -81 61 -44 -121 59 108 -107 -115 80 17 63 17 76 64 -94 -103 -55 -56 -80 0 36 113 -124 -39 16 -9 -38 -38 81 101 61 -80 119 -14 -29 120 -12 21 71 93 -12 -64 -92 123 60 -90 -118 -42 92 -98 113 -48 -39 -1 6 60 38 -115 24 44 81 27 15 77 -25 120 83 102 57 -92 -67 -56 14 -123 -102 -21 47 -28 83 -101 -74 -38 -80 -19 22 -121 -120 109 -53 67 14 33 33 -1 -32 15 -34 114 10 23 6 -22 53 117 -110 29 104 -117 124 -53 88 62 122 85 119 -87 -83 11 127 -64 -76 87 -107 -43 49 110 31 89 -37 -70 77 42 -109 23 -62 -33 -88 -13 32 -114 -116 -94 106 61 105 48 -3 82 -118 101 -69 -64 -81 -55 -21 39 2 11 -54 67 20 -123 -10 24 -123 -46 -34 89 111 22 -82 127 118 -105 32 0 -43 -23 69 -91 71 -126 -105 -63 50 -99 -74 -9 72 -8 -65 123 81 36 -79 93 116 -42 116 7 38 -67 -92 119 42 90 47 45 -75 1 -48 101 58 -38 -36 -117 58 127 62 118 76 -12 81 0 49 20 -74 -49 -17 -78 -46 106 49 102 59 123 123 -112 25 82 34 -106 111 37 -76 -55 -29 -115 -44 -96 56 -54 -88 -122 14 13 52 -5 85 -88 -89 -62 5 81 -97 96 -10 30 116 -38 55 44 122 -9 -75 104 58 -99 121 114 -126 -63 94 -9 -119 -71 -36 51 -125 -126 109 70 -76 -109 -70 -63 86 123 116 11 -111 48 -36 5 -98 -42 -104 -111]
(384 bilangan *signed 8-bit*)

Decrypted-text : Z1JYXF#1541955600#925.47#3572.55

Pencarian Nilai *Hash*

4 digit nol:

String:
`000000000000000000000000000000#QSQ4YE#1541955600#459.12#6521.56#9M1FO4WDNBYJQ1BE6L516AZZ17ZL5Z`

Nilai *hash*:
`0000D26BEB5824F55C8961DA84BECAC1EF7133877AF31C81AC68FFCEFC9EB802`

3 digit nol:

String:
`0000000000000000000000000000000#T5XOCV#1541955600#44.247#951.959#NRYE302203499G13Q11JV5450IMCX2`

Nilai *hash*:
000B5B89851DA6EDB6427C25D5CCA7600F189ADF002C737B1A29E1E060ADF56F

2 digit nol:
String:
00000000000000000000000000000000#L2CHZ6#1541955600#9.7117#77.2597#FQFC0CW9PVIS7AF6INF7FZJYKUJ1YL
Nilai *hash*:
00277A69E1630ADD640D031923827C75658ABB96A9339ED6F90428B2641E529B

Deretan 5 Blok Awal pada *Blockchain*
00000000000000000000000000000000#DXG56R#1541955600#96.98#359.198#QWRU39Z7LRH66V4H5QTLM6RDV2I579L
000EA081247BC62EBC3CC73C4C252CF8#ZSRFTA#1541955600#787.77#4462.34#ED6M2W0K3U0X205ON9042T8X41IP8Y
000BAA35A9F5FD15E07F19F16431D1C8#DLZF69#1541955600#7.5759#23.217#38SMAXSD4SR0L3UNQT1CAV95FUU9YIL
0001FD3950D0ECF2C540F01846F76B52#B6PH9Y#1541955600#678.07#3775.13#RRGQTWOERT4UMB6RR5MAU6G9X3439L
0004E212A56F29219CF6043E1ECE0BD2#ZN8S3F#1541955600#2.0865#34.2545#0AG4T1E6U3CMJJCH2R2EWYWKY3FBEW

LAMPIRAN E
DATA PENELITIAN

Pengujian Ukuran Kunci RSA
(baris pertama dalam satuan bit, baris lainnya dalam satuan detik)

1536	1600	1664	1728	1792	1856	1920	1984
0,2200	0,1227	0,3967	0,6504	0,4564	0,8194	0,4065	0,2611
0,0558	0,1353	0,0989	0,1831	0,3110	0,1803	0,1710	0,1470
0,2515	0,3299	0,1075	0,1344	0,1300	0,2998	0,2112	0,2150
0,1309	0,2653	0,1057	0,2932	0,5383	0,4139	0,6664	0,9515
0,2626	0,1407	0,4483	0,2373	0,1814	0,3568	0,2194	0,4929
0,4158	0,2402	0,1891	0,8491	0,3990	0,0762	0,2284	0,7510
0,1198	0,1258	0,1167	0,1751	0,5954	0,1378	0,4242	0,2956
0,1473	0,2349	0,5892	0,4086	0,0652	0,3220	0,0738	0,7501
0,0726	0,1808	0,2449	0,1817	0,3936	0,4454	0,3178	0,2402
0,2050	0,1804	0,1209	0,0745	0,3677	0,3897	0,1168	0,1974
0,1144	0,2170	0,1417	0,1403	0,0786	0,1981	0,2114	0,6582
0,0862	0,1145	0,1444	0,2282	0,6839	0,6275	0,1775	0,5317
0,1470	0,0644	0,2412	0,0900	0,3532	0,2330	0,3067	1,0185
0,1980	0,2708	0,2073	0,3610	0,2977	0,4173	0,3988	0,2643
0,1981	0,2823	0,1161	0,2411	0,1443	0,3436	0,1391	0,5626
0,1603	0,1776	0,2182	0,2562	0,2483	0,1451	0,2945	0,3511
0,1836	0,2595	0,2799	0,0677	0,1287	0,3784	0,3340	0,3791
0,1603	0,2432	0,1666	0,1031	0,1394	0,1652	0,1000	0,1784
0,1834	0,1838	0,1148	0,2182	0,1245	0,4235	0,2133	0,6271
0,2636	0,0634	0,4587	0,0759	0,2854	0,1153	0,8139	0,1792

2048	2112	2176	2240	2304	2368	2432	2496
0,4799	0,3037	0,4243	1,1080	0,4237	1,5348	0,2953	0,2097
1,0983	0,8636	0,2354	1,3019	1,2142	0,9853	1,1189	1,2539
0,2365	0,1491	0,3207	0,1897	0,4000	0,5312	0,1886	1,0284
0,6928	0,2492	1,0949	0,4194	0,4807	0,5523	0,3927	2,4880
0,1557	0,2504	0,6938	0,1827	0,4318	0,9103	0,3215	0,5749
0,4137	0,2757	0,6245	0,4533	0,3673	0,5310	0,2413	1,4674
0,7813	0,3066	0,8749	1,2403	0,7429	0,2682	1,1413	0,3196
0,4509	0,3006	0,6617	0,6099	0,3564	1,1636	0,2286	0,6165
0,5626	0,5901	1,1932	0,9744	1,0093	0,4831	0,3070	3,0896
0,4570	0,1239	0,1766	0,6044	0,3954	2,1099	2,5180	0,9880
0,6670	0,5280	0,5644	1,1583	0,7160	1,0174	2,4247	0,7279
0,2219	0,6740	0,2173	0,4299	1,0420	1,3559	0,8947	0,8101
0,1758	0,8663	0,5503	0,2118	0,4815	0,1449	0,3318	0,2216
0,8833	1,2001	0,0737	0,6031	0,7402	0,5270	1,0201	0,4398
0,4620	0,4594	0,6725	0,6973	0,6760	0,4383	0,7967	1,0954
0,1685	0,2191	1,1620	0,6557	2,5530	1,4292	0,5395	0,9198

0,1073	0,3449	0,4447	0,3012	0,4678	0,7025	0,2539	1,5582
0,4215	0,6241	0,1820	1,1035	1,1665	0,8237	0,9003	0,5587
0,4124	0,2395	0,3129	0,2974	0,4589	1,1431	1,1737	1,3391
0,6433	0,3452	0,3646	0,5708	0,2804	1,4444	0,3374	0,3716

2560	**2624**	**2688**	**2752**	**2816**	**2880**	**2944**	**3008**
1,3324	1,7172	0,4890	0,4345	1,4930	1,6940	0,9728	2,0649
1,3807	1,3379	0,9688	1,0852	0,8903	4,3057	0,7716	1,7764
0,1772	1,4599	1,8938	2,1347	0,5561	0,2973	2,6074	0,4210
1,8924	0,5310	1,9745	0,9394	0,2120	2,5511	1,3754	3,1620
0,8296	1,4068	0,7261	1,6213	1,3549	1,0268	0,9670	1,5772
2,1171	0,1741	0,3790	1,3079	0,5807	0,4799	1,6190	1,0342
1,4582	1,2885	0,7390	0,7860	2,6482	0,5169	0,7879	2,5351
1,6908	3,8064	0,6149	0,6116	1,8456	0,7103	3,1752	1,5885
0,3602	0,4177	0,9106	1,7772	0,4192	1,4560	0,6202	0,4370
0,1471	0,9382	1,0643	1,1926	0,4724	0,4262	0,2327	1,5364
1,1221	2,0872	1,1221	3,2818	1,0989	0,4458	2,2140	6,2538
0,2797	1,7588	0,4030	2,6908	1,1799	1,3441	2,7976	1,1793
1,1860	1,0121	0,8060	1,2006	0,8538	1,1758	0,9579	0,4440
0,4813	3,9890	0,5834	1,9914	0,8603	1,5156	3,6564	1,9133
0,6364	2,1782	0,4421	0,2470	2,7538	0,8033	0,9232	1,3412
0,3019	0,6167	1,0141	5,0403	0,5359	1,2222	0,6574	1,8723
1,3799	1,3145	1,5116	0,1583	1,3563	1,2538	4,0867	3,4727
1,0060	0,6892	1,2853	1,8050	1,0458	1,9201	2,6897	0,9677
1,4429	0,8182	1,0626	1,3560	0,8807	0,9162	2,2598	2,9465
0,3177	1,0250	0,6211	0,9246	0,6086	2,5668	3,2966	0,3427

3072	**3136**	**3200**	**3264**	**3328**	**3392**	**3456**	**3520**
0,8580	0,6901	1,2769	1,5366	1,7558	3,1418	1,0614	0,7465
1,9650	2,1422	4,6874	1,9922	2,8403	1,9486	0,9682	1,3535
0,8175	1,9383	2,1239	1,7898	4,0385	1,3855	0,3882	3,0462
1,5369	6,1278	1,5410	1,1101	1,4154	1,6802	4,1215	1,6434
4,2223	1,7798	0,4538	1,1711	0,7978	3,6825	6,9815	8,4928
1,6329	5,4715	1,9649	2,2571	1,6484	6,4872	2,3081	1,5082
1,7983	0,7906	3,9293	8,8043	1,9430	2,5825	2,0989	2,7899
3,3920	2,1200	1,9470	2,1793	0,8991	2,1928	1,0398	1,4442
1,2719	1,0983	1,3357	1,4079	1,1005	0,6361	2,8925	4,7302
0,4447	2,7485	1,3228	1,1078	1,0094	0,5836	5,7238	1,0845
1,5873	0,5083	2,3332	1,0069	0,9375	4,4795	4,6559	2,7910
1,7942	2,8726	0,7541	2,6844	1,2387	1,6314	1,5289	4,9280
1,0779	1,4566	3,7106	2,9456	2,0822	4,3696	0,5923	2,3585
1,8898	0,6204	3,7575	1,8476	2,0424	4,8284	2,8585	2,1259
1,3469	2,1938	3,8166	1,5462	5,5553	4,4221	4,0829	1,9158
3,6299	1,7744	3,0304	1,0429	2,5910	5,7327	4,0739	5,2482
1,0585	3,8080	0,3805	0,8251	5,9560	4,0638	2,1286	3,2463
1,4846	1,9175	1,7650	4,5156	2,2275	2,5058	9,4265	2,9006

0,6726	3,5705	0,8348	3,7440	2,6699	3,4317	2,5127	4,2208
3,6986	1,0285	5,4148	1,3189	1,7499	0,7998	0,6530	1,8790

3584	**3648**	**3712**	**3776**	**3840**
2,0411	6,6699	2,0008	5,4782	1,9534
7,8352	2,3799	4,9269	4,5071	2,7200
1,6625	1,9030	4,4916	5,8713	4,1738
4,9088	9,3831	5,1690	1,5711	4,8045
2,7782	4,1702	0,8197	1,1576	7,9189
3,1496	2,6209	3,5483	1,2012	4,2868
4,9188	2,0024	3,6772	6,3743	1,9867
1,9766	1,9748	2,6582	2,3674	7,0275
1,5953	3,8428	1,9032	2,2039	3,6149
0,5829	4,1920	1,3293	3,1661	5,2288
2,2930	3,0556	2,0026	3,8943	1,7545
4,9873	5,5602	2,9732	6,3306	4,4405
1,9912	3,4267	1,0767	1,6230	3,0072
4,0567	1,1129	2,4846	3,2799	7,5094
4,0335	1,4218	1,4846	4,1841	3,5928
11,1963	2,6553	2,1057	1,9802	2,4284
0,5915	6,3074	4,6746	2,2687	3,5705
0,4325	3,7601	2,9880	2,1254	2,0011
4,8427	1,2841	1,4511	7,0777	4,5320
3,4482	1,9027	2,1488	2,5479	0,8120

3904	**3968**	**4032**	**4096**
7,7449	1,8399	3,3338	6,4928
2,0712	11,3981	10,2019	5,6891
1,4592	2,5483	4,0693	2,2079
6,1066	5,7505	1,8900	1,6194
2,5657	5,1090	2,5549	6,6757
5,5616	8,5238	9,7219	14,5828
1,6676	1,9999	2,6485	7,2293
10,2356	0,7643	14,2126	6,3218
8,2940	7,1477	3,9636	11,0519
1,5365	10,3767	7,4724	3,4907
5,4218	3,6519	1,5581	7,1192
5,1181	3,9022	4,2799	6,6474
3,5001	2,3036	6,3427	6,7082
5,5209	1,3666	9,2655	2,4431
4,7489	10,9405	3,8499	13,5793
3,2123	6,3727	5,8705	1,3840
3,4344	2,8241	3,2293	4,5007
0,7529	0,4651	3,7454	5,1716
2,3374	5,1775	3,8355	1,4817
4,7260	1,8573	10,6811	5,3518

Rata-rata Waktu Komputasi Setiap Ukuran Kunci RSA

1536	1600	1664	1728	1792	1856	1920	1984
0,1788	0,1916	0,2253	0,2485	0,2961	0,3244	0,2912	0,4526

2048	2112	2176	2240	2304	2368	2432	2496
0,4746	0,4457	0,5422	0,6557	0,7202	0,9048	0,7713	1,0039

2560	2624	2688	2752	2816	2880	2944	3008
0,9770	1,4283	0,9306	1,5293	1,0823	1,3314	1,8334	1,8433

3072	3136	3200	3264	3328	3392	3456	3520
1,8090	2,2329	2,3190	2,2417	2,2249	3,0293	3,0048	2,9227

3584	3648	3712	3776	3840
3,4661	3,4813	2,6957	3,4605	3,8682

3904	3968	4032	4096
4,3008	4,7160	5,6363	5,9874

(satuan bit)
(satuan detik)

Pencarian Nilai Hash (angka dalam satuan detik)
2 Digit Nol

0,2560	0,1974	0,0619	0,0222	0,2442	0,1431
0,1432	0,1482	0,0401	0,0896	0,0897	0,1324
0,0367	0,1501	0,1237	0,0591	0,0081	0,0195
0,0613	0,1185	0,1069	0,2952	0,1036	0,1062
0,0841	0,1411	0,0295	0,1524	0,0371	0,1865
0,0046	0,1152	0,0956	0,0203	0,0144	0,1922
0,0410	0,1470	0,1359	0,0363	0,0426	0,0845
0,0078	0,0301	0,0054	0,0413	0,0925	0,0943
0,0140	0,1628	0,1300	0,1827	0,1283	0,0997
0,0923	0,1502	0,1128	0,0192	0,1519	0,1555
0,3287	0,0074	0,0604	0,0083	0,0014	0,0311
0,0196	0,0020	0,0562	0,1104	0,1035	0,0698
0,0836	0,0344	0,0220	0,0145	0,0681	0,0162
0,1019	0,2476	0,2374	0,0857	0,0719	0,1493
0,0907	0,0064	0,2404	0,0041	0,3852	0,0677
0,0831	0,2207	0,1439	0,0857	0,0214	0,0725
0,0092	0,0178	0,0442	0,0275	0,0342	0,1048
0,0048	0,0715	0,0111	0,0880	0,1205	0,1022
0,0179	0,0932	0,2952	0,2099	0,0987	0,0098
0,0678	0,0544	0,0930	0,0966	0,0152	0,1674
0,0203	0,1194	0,2837	0,0471	0,0207	0,0815

0,0443	0,0552	0,4568	0,3939	0,0647	0,0281
0,0155	0,0757	0,3005	0,3336	0,1086	0,1487
0,0079	0,0641	0,0248	0,0553	0,1104	0,2244
0,0119	0,0042	0,0350	0,0401	0,0340	0,0217
0,1544	0,0450	0,0746	0,2145	0,0722	0,3009
0,0015	0,1939	0,0021	0,0148	0,0626	0,1960
0,1605	0,0790	0,3405	0,0768	0,0529	0,1366
0,0089	0,1546	0,0615	0,0324	0,0509	0,1214
0,0270	0,0434	0,0287	0,2085	0,0566	0,0442
0,0529	0,0111	0,0500	0,0910	0,0997	0,0011
0,2728	0,0379	0,1278	0,0369	0,0772	0,0811
0,0551	0,0133	0,0333	0,1236	0,0012	0,0175
0,0457	0,0226	0,0590	0,1082	0,0359	0,4108
0,1747	0,0049	0,0551	0,0162	0,0103	0,0695
0,0063	0,1001	0,1191	0,0701	0,4585	0,0495
0,1086	0,0504	0,0018	0,0366	0,1923	0,1441
0,0854	0,1445	0,0042	0,2225	0,1228	0,1056
0,0803	0,1643	0,0440	0,3842	0,0078	0,0767
0,0275	0,0119	0,1314	0,0925	0,1212	0,0108
0,1215	0,0603	0,0121	0,0315	0,0768	
0,1916	0,2444	0,0463	0,0606	0,3640	

3 Digit Nol

0,9681	3,1566	2,3019	2,4954	4,5469	1,2033
2,7948	1,5825	1,9145	2,7195	0,1709	4,4074
0,0665	0,2967	0,3638	1,0977	2,5472	1,4956
6,5526	1,2665	0,5285	1,3575	1,5204	0,4309
0,4310	3,7971	4,5995	0,1342	1,8244	1,1125
2,6438	0,9830	3,0738	4,7061	0,1363	0,1072
0,5794	1,1378	5,8288	1,9997	1,0480	1,4142
4,2129	0,3395	0,3784	1,1594	2,4600	0,0989
0,0077	1,7487	1,3217	0,8963	0,1933	4,5183
3,4644	1,7708	0,0148	0,5527	0,4944	1,4045
1,9951	0,3911	0,1407	7,7520	0,0293	2,9522
0,8511	0,0563	0,4572	3,0600	0,4807	0,5477
0,9138	0,1041	2,7640	0,4959	1,0073	0,7323
0,2211	0,0831	1,7351	1,1396	0,6916	0,7811
1,7603	2,0321	1,0104	2,1999	0,4421	0,4763
1,4224	0,7685	0,1913	5,6685	2,4168	1,0959
0,2704	3,1797	1,0063	0,0394	0,2046	3,4589
1,4488	1,6525	2,9947	0,6177	0,8121	2,3752
0,0479	1,0694	5,4426	3,2410	0,1728	5,5504
0,2503	0,3058	0,4760	0,8199	0,6060	0,4332
2,4508	0,0786	1,1162	3,7046	1,6203	1,6086
0,0121	1,2174	0,3184	0,5971	0,2871	2,8437
1,7083	1,2888	5,4174	0,0022	3,0089	0,5276

1,9009	0,2973	1,1760	0,5075	5,9848	0,1968
2,1161	0,7268	0,1812	0,9641	0,9847	0,0589
1,5016	0,2458	3,0045	0,4257	1,2922	0,6030
3,2903	2,5748	0,2369	0,9990	2,0084	0,7653
1,8923	1,2865	0,5405	0,2705	0,7800	4,1392
0,2679	4,5864	0,9046	2,4861	0,7890	0,4087
1,3029	0,0162	3,7562	0,5954	0,7727	1,3745
2,9525	2,1185	0,2753	7,1900	2,6056	1,4508
1,2824	3,9881	1,2563	0,1098	0,7846	1,9602
0,2365	0,0205	4,8604	0,5101	2,1154	0,7282
2,4670	0,2888	3,3305	0,1114	0,4359	0,0376
0,1596	5,7512	2,1798	0,7342	0,0515	0,6668
0,1269	0,8666	0,6890	2,5055	0,7991	3,7784
0,9524	0,6673	1,4771	2,6006	0,5949	1,0551
1,9496	0,3575	0,4406	0,4119	0,0008	0,1676
0,9703	0,2740	0,8120	5,0966	0,9679	2,4655
4,0628	2,1446	2,1956	3,0793	0,5492	2,6926
1,3580	0,5123	1,9645	1,2780	0,1500	
0,1316	0,3721	2,8780	0,1930	0,8022	

4 Digit Nol (Datum *outlier* tertanda)

4,7025	10,3380	23,1154	16,3462	63,5633	50,4394
13,9086	25,4513	54,1857	1,8938	2,2583	13,8635
4,8605	35,4298	7,0122	29,0211	15,5826	7,5101
27,1781	9,5809	7,3101	6,5874	31,3629	15,6260
11,5100	13,1403	46,9907	19,8997	3,3274	35,1899
9,6373	58,8514	38,0373	1,8357	2,5432	19,1116
2,7935	5,1894	22,8494	13,9888	1,6678	22,7643
11,3304	15,6812	1,5096	33,9638	0,6975	7,8277
10,4707	13,0167	19,2597	5,7855	1,9652	47,9262
71,2369	25,4091	46,9604	22,3349	46,0519	9,9051
53,3060	7,0334	76,0392	4,7571	6,8111	33,4814
22,4110	2,5740	12,1295	18,9396	10,1074	9,3349
8,4913	26,0935	4,0667	9,7791	62,8547	2,8999
1,3616	4,1686	12,8809	15,3142	27,2453	59,8698
15,1685	12,9553	9,4510	19,2302	27,3243	17,7651
17,9498	46,0249	28,4321	2,9969	54,3043	52,6578
16,8023	16,7956	77,0732	3,3708	53,7052	26,6032
56,5862	31,4612	15,7529	135,0322	32,6593	0,9161
16,6729	59,3730	6,8567	1,6899	26,6269	37,1212
42,6580	2,8217	9,2336	10,3476	3,9230	34,1582
24,8050	16,9589	9,1872	25,2762	37,3678	38,9048
1,4993	33,9250	29,4728	5,6659	23,1735	33,0901
4,0574	5,5462	32,0399	19,6656	7,1369	11,0007
12,7184	3,7465	6,5631	44,4222	79,6073	5,8569
41,6322	4,1719	17,0659	28,5458	30,0235	2,7851

1,3820	13,6887	7,9464	15,6248	8,7491	1,0578
7,4293	39,7827	9,0478	4,0685	49,0307	4,8297
35,5898	29,6167	9,7412	18,9221	32,9214	1,9622
76,8100	31,1932	2,8178	21,9427	36,0693	24,2393
9,5247	38,7192	3,9125	4,0785	40,4511	22,1115
18,4636	10,4677	24,5628	3,7658	13,1452	52,7828
43,6328	60,5380	23,7209	24,8591	51,8701	12,6719
14,5835	9,9343	19,3352	68,7286	9,3976	37,1107
9,2888	8,0201	19,7454	4,6394	30,7672	4,3338
14,0569	29,9211	21,7317	15,5150	23,3427	8,4917
8,7286	1,7886	18,0104	2,2854	13,7312	10,1996
28,4339	8,4142	29,9455	14,8451	36,7278	2,1191
22,6210	29,0238	1,3410	41,7888	6,0446	12,7355
14,8860	3,7549	21,0624	32,4621	43,0938	2,2838
29,9039	8,5950	15,0356	38,2210	0,4703	44,6425
21,9037	25,0666	3,4652	33,0568	46,5203	
137,1826	16,8697	15,2923	2,6034	10,1793	

Simulasi Program RSA pada Jaringan *Smart Grid*
(angka dalam satuan detik)

12,1217	11,5181	12,2702	10,6454	10,3790	17,7224
11,0429	9,9015	9,9141	12,0167	10,4942	19,2296
11,0684	12,9274	9,2293	12,3014	14,6547	11,0446
9,8646	14,4889	11,5834	10,7067	12,7997	14,4637
13,5025	13,4210	11,1018	18,2127	12,3552	10,5237
11,1445	11,1662	13,3667	11,5167	9,9010	14,9313
10,5678	14,0919	11,0944	13,5854	11,4095	12,1627
8,0604	13,4104	11,7919	15,0568	10,6860	15,0289
14,0427	13,9028	11,5391	11,8017	9,8921	14,3639
12,9876	12,4860	13,8809	10,0543	13,8664	8,7920
13,3692	14,4496	10,8973	10,4151	12,6324	10,6726
19,2570	10,2023	12,6298	12,9098	10,9284	10,7514
12,8702	14,7032	13,6290	11,6425	14,6409	12,1287
11,7277	11,8626	12,4298	13,0847	10,7213	12,7804
11,9467	11,7177	13,9371	10,8310	11,1719	9,4277
13,5471	11,0591	12,1738	14,1275	17,0307	12,8742
18,0225	13,1065	10,3393	9,8782	13,9269	13,1696
11,6003	12,4953	12,9776	10,8028	13,9566	11,3178
12,3501	10,8478	11,0568	9,8296	14,5477	15,6651
11,1414	13,6459	12,7442	13,1858	11,1194	10,8479
9,8774	10,2237	9,7248	10,4265	12,1730	15,6885
14,5389	13,6923	11,4900	12,8462	11,7360	12,8097
9,5422	13,2255	10,3167	11,2842	12,1891	10,0488
11,1519	9,1747	15,0105	11,3615	13,2283	13,2647

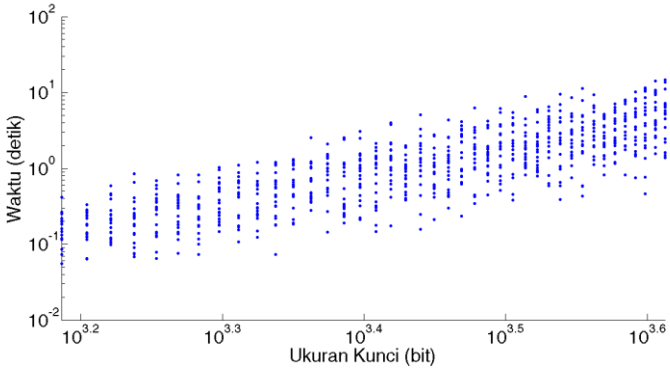
10,6474	9,9928	11,9627	13,7186	12,5450	11,6342
12,7966	14,7404	14,6464	10,5369	13,4808	14,5587
10,3895	11,2170	13,6099	11,2734	10,9928	11,5490
12,3874	13,2547	10,8264	11,6945	12,0546	13,3499
10,8139	8,1030	10,8441	12,0139	10,2537	11,2372
11,9562	12,4366	10,4926	12,5357	11,2399	11,9625
13,8619	9,2055	12,6414	10,7488	9,1664	10,9835
11,5959	12,1673	13,3001	11,2152	11,6989	14,3130
10,2105	10,1511	10,5934	10,8767	11,6392	9,0901
9,9149	11,2857	10,4906	10,3142	10,3852	9,6028
11,2441	12,7901	12,1921	15,2806	11,8085	10,5570
12,1892	10,9096	15,5241	12,9905	13,2700	13,0615
13,3681	10,8298	10,1189	10,4177	12,2127	12,2590
9,7415	13,7803	11,5058	9,9377	10,8765	11,4955
11,5743	12,8691	12,2703	9,2021	9,1784	12,7125
12,5070	10,6153	11,5530	10,5789	12,7059	13,6653
9,3767	11,0589	18,1473	9,4495	15,5503	
13,7999	12,9696	9,9626	9,0092	14,8797	

Simulasi Program *Blockchain* pada Jaringan *Smart Grid* (angka dalam satuan detik)

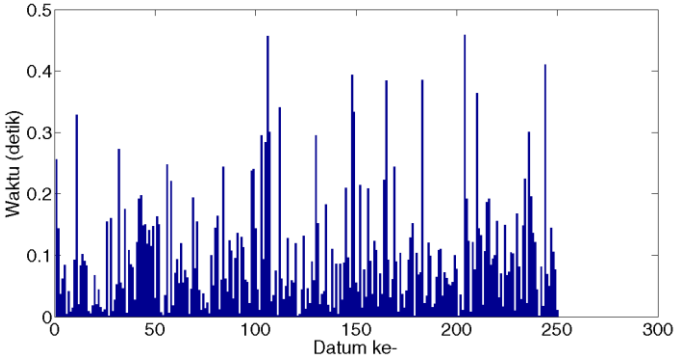
66,7676	80,7667	87,2526	71,2502	70,4483	74,3063
56,4208	83,0602	86,5202	92,4350	71,0115	63,2725
82,2519	88,7710	87,2544	77,7907	58,0952	66,6785
58,1341	74,3682	69,6045	68,9241	66,6100	69,5540
76,2015	89,2088	72,0619	75,2875	91,3949	69,3109
72,7872	56,9827	70,9356	91,9041	65,2603	
71,8793	68,4387	64,0683	64,7210	83,9752	
75,3340	75,4021	71,0860	59,8678	76,1223	
88,3913	91,6428	75,3188	96,3993	65,0863	

LAMPIRAN F
TAMBAHAN GRAFIK DATA PENELITIAN

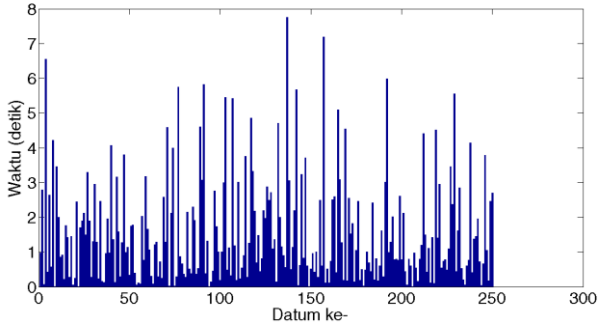
Program RSA Individual (Log-log)



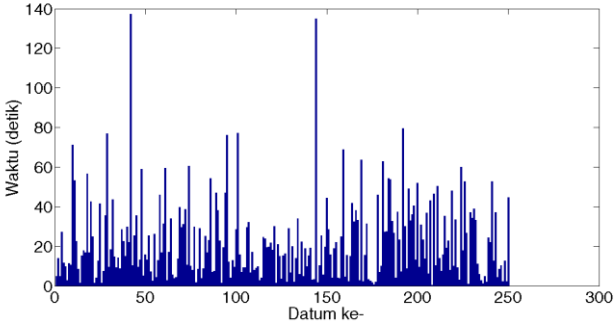
Program Pencarian Nilai *Hash*
2 Digit Nol



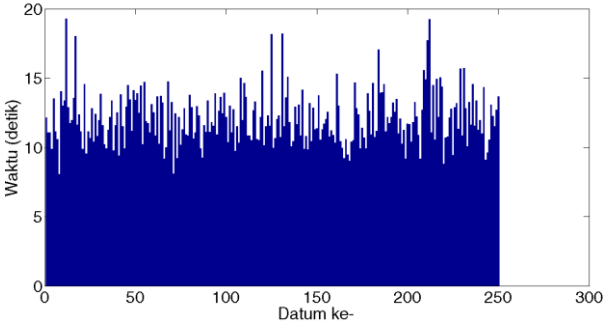
3 Digit Nol



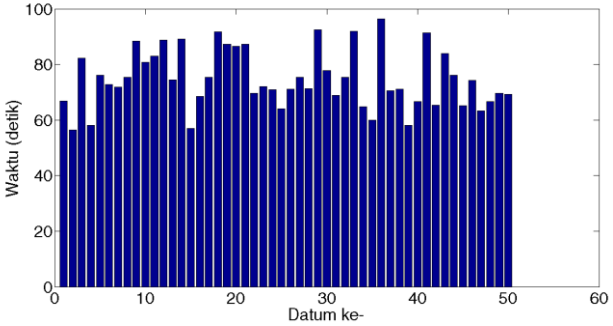
4 Digit Nol



Program RSA pada Jaringan *Smart Grid*



Program *Blockchain* pada Jaringan *Smart Grid*



Halaman ini sengaja dikosongkan

BIODATA PENULIS

Hafizh Fianto Putra. Lahir di Kota Mojokerto pada hari Selasa tanggal 12 November 1996. Penulis biasa dipanggil Hafizh atau Putra. Penulis merupakan anak pertama dan satu-satunya laki-laki dari tiga bersaudara. Saat buku ini ditulis, ayah penulis yang bernama Sunu Hartanto bekerja di PT Waskita Karya, sedangkan ibu penulis yang bernama Fitri Prastajani menjadi ibu rumah tangga.

Alasan utama penulis menjadi mahasiswa departemen Teknik Elektro ITS adalah penulis ingin meneruskan jejak bapaknya yang juga merupakan alumni ITS. Selain itu, penulis juga memiliki ketertarikan pada bidang TIK serta kemampuan lebih di bidang matematika dan fisika. Meskipun sempat tidak diterima melalui jalur undangan di jurusan Teknik Multimedia dan Jaringan (sekarang departemen Teknik Komputer) ITS, penulis tidak putus asa dan mencoba kembali melalui jalur tes hingga akhirnya penulis diterima di tahun 2014, tahun yang sama dengan kelulusannya dari SMA Negeri 2 Kota Mojokerto.

Selama menjadi mahasiswa, banyak hal yang telah dialami oleh penulis. Salah satunya yang paling berharga adalah kesempatan untuk mengikuti pertukaran pelajar di negara Jepang selama satu semester. Sebagai salah satu pecinta budaya Jepang modern seperti *anime* dan musik pop Jepang (J-pop), penulis tidak menyia-nyiakan waktunya selama menjadi mahasiswa di Kumamoto University. Penulis fokus untuk melatih kemampuannya berbahasa Jepang selama pertukaran pelajar, sehingga penulis memilih untuk menunda kelulusannya di ITS. Namun, penulis tetap merasa senang dengan banyaknya pengalaman pertama yang bisa didapatkan di Jepang, seperti: merasakan salju, menikmati pemandian air panas, menjadi pegawai di hotel, dan melihat secara langsung referensi tempat di *anime*. Setelah pertukaran pelajar, penulis semakin bersemangat untuk bisa melanjutkan studi S2-nya di Jepang.

Cita-cita penulis adalah menjadi dosen di universitas terkemuka di Indonesia seperti ITS. Namun, penulis tidak menutup kemungkinan untuk bisa bekerja di tingkat internasional. Penulis ingin bisa membalas segala kebaikan orang tuanya yang telah membantunya. Penulis juga ingin membuat keduanya bangga atas keberhasilan penulis. Bagi penulis, tidak ada yang tidak mungkin jika Allah SWT menghendaki-Nya.

Untuk informasi lebih lanjut,
penulis dapat dihubungi melalui
Facebook atau alamat surel
hafizhfp@gmail.com